Just Simple Sequencing

Learning Systems Architecture Laboratory
Carnegie Mellon University
Pittsburgh, PA 15213 USA
Web: http://www.lsal.cmu.edu/
Email: lsal@cmu.edu

August 2002
Just Simple Sequencing

1. Introduction

The IMS Simple Sequencing Specification (SS) has a sound basis although other underlying models are equally applicable. The development of the draft specification from the original basis has resulted in a set of documents that are recognized to contain deficiencies (presented in no particular order):

- The documents contain repetitive text and descriptions.
- Duplicative descriptions are in conflict.
- Behavior and data descriptions are intermixed: data model descriptions include behaviors and behaviors include data descriptions.
- Key behaviors and processes used to implement SS, such as handling error conditions and exceptions, are not specified.
- Certain behaviors and processes used to implement SS are only partially specified.
- There are undocumented assumptions and behaviors.
- SS includes features that may have limited utility and extend beyond the intent of “simple” sequencing.
- SS contains features that overlap with and may be in conflict with other learning technology specifications.
- Some descriptions and terminology are confusing, especially to instructional designers and content authors.
- The terminology and jargon assume a high level of familiarity with the existing CMI and SCORM models.
- SS contains application profile–specific features.
- While conceived as a model of sequencing, SS includes other features that are not strictly sequencing and properly belong in other specifications.
- There are implicit assumptions that violate the distinction between sequencing behaviors and presentation of navigation elements.
- While designed to be content-type agnostic, SS is biased towards content that uses the API or an equivalent communications mechanism.
- The behaviors are not fully consistent with the underlying model.

As a result, many of the developers of SS do not feel that even they fully understand all that it contains and implies. It is critical that the problems with SS be remedied before it becomes a Final IMS Specification. As written, different implementations that are consistent with both the letter and spirit of SS would produce different learning experiences. SS does not satisfy the key goal of interoperability of learning experiences across different systems.

This document set is an alternative presentation of the SS – Just Simple Sequencing. It addresses only the core technical specification, not the binding, conceptual model, use case or best practice guide portions of the full SS document set. The key aspects of this alternative presentation are:

- It attempts to be identical in functionality and intent with that described in the IMS SS Public Draft.
- Descriptions and terminology have been simplified to improve readability.
- Notations and tokens describing data element names have been changed in this description, but these changes do not necessarily apply to the binding.
- The information has been reorganized by functional category and duplicative content has been eliminated.
- References to application profile–specific behaviors and other descriptions not pertaining to just simple sequencing have been eliminated.
- Data items that are not related to sequencing or that are otherwise incomplete have been eliminated.
Ambiguous and incomplete behaviors have been detailed and open issues have been identified.

The description is based solely on the IMS SS Public Draft. Implicit or missing behaviors not included in the IMS SS Public Draft have not been added to this version. Interpretations and revisions are based only on the descriptions and behaviors included in the Specification part of the document set.

2. Specification Elements

This presentation is divided into several sections, each section being standalone and describing one part of the sequencing specification.

- **Sequencing Definition Model** – The overall data model in SS used to describe the desired sequencing behavior (see part SM).
- **Tracking Model** – The data model in SS used to record information about the results of a learner’s interactions with activities, and the learner’s record for objectives (e.g., completion, measure) to control the selection and sequencing of other activities (see part TM).
- **Activity State Model** – The data model in SS used to record information about the state or status of a learner’s interactions with an activity and a set of global attributes for activities (see part AM).
- **Navigation Behavior Model** – The SS process that evaluates a navigation request and determines the sequencing and exit requests that should be processed to identify and deliver content to the learner (see part NB).
- **Exit Behavior Model** – The SS process that evaluates an exit request to terminate a content activity (see part EB).
- **Sequencing Behavior Model** – The SS process that evaluates a sequencing request in terms of the content model described by the activity tree and determines what actual content object should be delivered to the learner (see part SB).
- **Delivery Behavior Model** – The SS process that validates that the content resources for the identified activity may be delivered, i.e., all of the conditions that apply to the delivery of the content for the activity and attempt still hold (see part DB).
- **Rollup Behavior Model** – The SS process that computes the results data for an activity from the results data from the children of the activity (see part RB).
- **Overall Sequencing Process** – The overall SS process that relates the navigation, sequencing, delivery, and rollup processes (see part OP).
- **Extensibility** – The extensibility mechanisms in SS (see part E).
- **Conformance** – Requirements on an implementation to conform to the SS specification (see part C).
- **Glossary** – A collection of terms used throughout the SS document set to describe parts of the data models of sequencing behaviors specific to this description of SS (see part G).

Each piece follows as an independent section. The sections are intentionally independent so that an implementer or reader has all the information related to the topic and other key information needed to interpret and implement the corresponding part of a SS system. Each section includes:

- **Information Model** – The detailed description of the corresponding information model (for the sequencing definition model, the tracking model, and the activity state model). Each information model is divided into logically related sets of data attributes. Each set describes the data items in a binding-independent fashion.
- **Behavior** – The detailed description of the behaviors and procedures that describe the parts of SS (for each of the behavior models). Each behavior includes one or more process descriptions in pseudo code.
- **Notes/Changes from IMS SS Public Draft** – How the elements of the part of the model differ from the IMS Public Draft Simple Sequencing Specification (version 0.7.5, April 12, 2002), including changes, deletions and additions.
- **Open Issues** – Specific problems with the SS specification that require additional resolutions within the IMS community.
• **Implementation Recommendations** – Suggestions and recommendations on how to implement SS. These are not best practices, as there currently are no best practices, but rather specific implementation hints and details that are not included in the technical specification of SS.

• **Bindings** – An overview of how the information model is bound to an XML representation for interchange within an IMS Content Package.

• **Service Descriptions** – Behaviors and processes described as services, with requests, processing and response descriptions for each service access point. Service descriptions provide the basis for implementing SS as a service-based system consistent with the SS conceptual model.

This organization is designed to present the parts of the model and the related items and issues in a way that may also focus discussion on resolving issues. This organization is not optimal for presenting SS to a larger developer community.

3. **Notes**

Elements of this document are extracted from the IMS Public Draft Simple Sequencing Specification (version 0.7.5, April 12, 2002), © Copyright 2002, IMS Global Learning Consortium.

It is not intended that this introductory section be included in any revised IMS document set. It is also not intended that the detailed sections be used verbatim in any IMS document.
SM. Sequencing Definition Model (Just Definition V1.0)

The SS process uses information about the desired sequencing behavior to control the sequencing, selection and delivery of activities to the learner. The intended sequence is described by a specific set of data attributes. These attributes are associated with learning activities in the activity tree to describe the sequencing behavior. The defined set of attributes used by SS is called the “sequencing definition model”.

The sequencing definition model consists of:

- **Sequencing Control Modes** – controls for types of sequencing requests that may apply to a collection of activities.
- **Sequencing Rule Definitions** – rules applied to an activity used to specify sequencing behaviors for the activity.
- **Limit Conditions** – limits on how many times, how long and when an activity is available.
- **Sequencing Thread Definitions** – specifications of parallel activity threads.
- **Rollup Rule Definitions** – rules that specify how tracking data from an activity or its associated objectives is produced from the results of the child activities.
- **Objective Description** – the learning objective associated with an activity.
- **Rollup Controls** – controls which data from an activity contributes to the rollup of the parent activity.
- **Delivery Controls** – controls when progress and objective data for an activity are recorded.

There are no overall behavior requirements on the use and instantiation of the sequencing definition model. Individual parts of the model describe how the sequencing definitions are associated with the activity tree to define intended learning experiences. The use of the sequencing definition model is detailed as part of the behavior descriptions of the navigation, sequencing, delivery, exit, and rollup processes.

Sequencing definitions describe the intended learning experience for a learner. How the learning experience and definition for a learner relate to the definition for a cohort of learners (e.g., individualized experiences versus a common experience for all members of the cohort) is not specified. A sequencing definition model instance defines a learning experience independently of how it is instantiated for one or more learners.

SM.1 Sequencing Definition Model

The sequencing definition model describes the data that specifies the intended sequencing behavior. This data is used by a system that delivers sequenced activities. How this information is encoded, stored, represented or bound is outside the scope of this specification. The sequencing definition model only describes a set of related data items and internal constraints on the values of those items.

There is no requirement that the value for any specific sequencing definition model data item exist. There is a defined default or initial value for each attribute. The default value is supplied upon reference if the data value does not exist.

The mechanisms used to create or maintain the sequencing definition model data are not specified as part of the information model.

An implementation must be capable of representing the range of values described. There are no additional requirements on implementing the sequencing definition model.

The sequencing behavior model and delivery behavior model (see Sequencing Behavior Document SB, Deliery Behavior Document DB) describes how a sequencing system uses a sequencing definition instance for an activity tree to control the sequencing and delivery of activities.
SM.1.1 Sequencing Control Modes

Sequencing control mode information (the set of attributes shown below) includes descriptions of the types of sequencing behaviors specified for an activity. Sequencing control mode information for an activity includes the associated data listed below.

SS processes may reference the sequencing control mode data for any activity in the activity tree. Sequencing control mode data need not be instantiated for each activity in the activity tree. Data for parent activities is used if data is not available for an activity.

The control modes are not exclusive; multiple control modes may be specified.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Description</th>
<th>Value Space</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sequencing Control</td>
<td>Indicates that a Choice sequencing request is permitted (True or False) for</td>
<td>Boolean</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>Choice</td>
<td>the children of the activity.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Sequencing Control</td>
<td>Indicates that a Flow sequencing request is permitted (True or False) for</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Flow</td>
<td>the children of the activity.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Sequencing Control</td>
<td>Indicates that a Continue sequencing request is generated (True or False)</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Auto Advance</td>
<td>for the children of the activity when the child activity finishes. The</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>mechanism for generating the request is not specified in this model. The</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>mechanism for determining when the child activity is finished is specified</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>in the SS behavior models.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Sequencing Control</td>
<td>Indicates that backward targets (in terms of activity tree traversal) of</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Forward Only</td>
<td>Choice and Flow/Previous sequencing requests are permitted (True or False)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>for the children of the activity.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SM.1.2 Sequencing Rule Descriptions

Sequencing rule descriptions (the set of attributes shown below) specify the details of individual rule-based sequencing behaviors for an activity. Sequencing rule description information for an activity includes the associated data listed below.

SS processes may reference the sequencing rules for any activity in the activity tree. Sequencing rules are instantiated only where needed. Default data is used if the data is not instantiated for the activity.

The data attributes describe one rule. Each activity may have an unlimited number of unordered sequencing rules.

The general format of a rule can be expressed informally as: If conditions Then action.

- There are multiple conditions.
- The conditions may be combined with a single and combination (all conditions must be True) or a single or combination (only one condition must be True).
- Individual condition values may be negated before being combined in the rule evaluation.
- No other operators on the conditions are defined.
- Each condition references an item in the tracking model for the activity or its associated objective.
- There is one action that may result if the rule conditions evaluate to True.

Actions are divided into two groups:

- Actions that control sequencing decisions and delivery of a specific activity. Rules that include such actions are used to determine if the activity will be delivered.
- Actions that control sequencing flow by issuing exit requests. Rules that include such actions are applied when an activity completes.
The Rule Conditions and Rule Actions are tokens in a vocabulary. The tokens have no semantics or meanings themselves. The description of the condition or action is the complete definition of the required behavior.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Description</th>
<th>Value Space</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rule Combination</td>
<td>How rule conditions are combined in evaluating the rule.</td>
<td>Vocabulary</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <em>All</em> – The rule condition evaluates to True if and only if all of the individual rule conditions evaluate to True (logical <em>and</em>).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <em>Any</em> – The rule condition evaluations to True if any of the individual rule conditions evaluates to True (logical <em>or</em>).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Rule Conditions</td>
<td>An unordered collection of conditions for a sequencing rule. A rule includes multiple rule conditions (1:N)</td>
<td>Collection 1:N</td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>Rule Condition</td>
<td>A condition element for the rule.</td>
<td>Vocabulary</td>
<td>Always</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <em>Satisfied</em> – evaluates to True if the Objective value of Objective Satisfied Status for the objective associated with the activity (indicated by Objective ID) is True.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <em>Not Satisfied</em> – evaluates to True if the Objective value of Objective Satisfied Status for the objective associated with the activity (indicated by Objective ID) is false.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <em>Objective Status Known</em> – evaluates to True if the Objective value of Objective Data Status for the objective associated with the activity (indicated by Objective ID) is True.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <em>Completed</em> – evaluates to True if the Activity Progress value of Activity Completion Amount for the activity is 1.0 (i.e., the activity is complete).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <em>Incomplete</em> – evaluates to True if the Activity Progress value of Activity Completion Amount for the activity is not 1.0 (i.e., the activity is incomplete).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <em>Activity Progress Known</em> – evaluates to True if the Activity Progress value of Activity Completion Status for the activity is True.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <em>Attempted</em> – evaluates to True if the Activity Progress value of Activity Attempt Count for the activity is positive (i.e., the activity has been attempted).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <em>Attempt Limit Exceeded</em> – evaluates to True if the Activity Progress value of Activity Attempt Count for the activity is equal to or greater than the Limit Condition value of Limit Condition Attempt Limit for the activity.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <em>Time Limit Exceeded</em> – evaluates to True if any of the Activity Progress duration values for the activity (Activity WallClock Duration, Activity Interaction Duration, Activity Attempt WallClock Duration, Activity Attempt Interaction Duration) exceed the corresponding duration Limit Condition values for the activity (Activity WallClock Duration Limit, Activity Interaction Duration Limit, Activity Attempt WallClock Duration Limit, Activity Attempt Interaction Duration Limit).</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Interaction Duration Limit

- **Outside Available Time Range** – evaluates to True if the current time is before or after the corresponding time Limit Conditions for the activity. (*Begin Time Limit, End Time Limit*).
- **Always** – always evaluates to True.

### Rule Condition Operator

The unary logical operator to be applied to a condition.

- **Not** – The corresponding condition is negated in rule evaluation.
- **NO-OP** – The corresponding condition is used as is in rule evaluation.

### Rule Action

The desired sequencing behavior if the rule evaluates to True.

Actions are divided into groups based on when the action is applied.

**Precondition actions applied when traversing the activity tree to select an activity for delivery:**

- **Skip** – Skip the activity when traversing the activity tree while processing a sequencing request.
- **OK** – The rule causes no action.
- **Disabled** – The activity may not be the target of any sequencing or delivery request.
- **Hidden from Choice** – The activity may not be the target of a “choice” sequencing request.
- **Stop Forward Traversal** – Stop walking the activity tree at the current activity while processing a sequencing request.

**Postcondition actions applied when the activity attempt terminates:**

- **Exit Parent** – Change the exit and navigation requests to *Exit Parent*.
- **Exit All** – Change the exit and navigation requests to *Exit All*.
- **Retry** – Change the exit and navigation requests to *Retry*.
- **Retry All** – Change the exit and navigation requests to *Retry All*.
- **Continue** – Change the exit and navigation requests to *Continue*.
- **Continue From Parent** – Change the exit and navigation requests to *Continue from Parent*.
- **Previous** – Change the exit and navigation requests to *Previous*.
- **Ignore** – No action.

### SM.1.3 Limit Conditions

Limit conditions (the set of attributes shown below) define constraints on the access to an activity based on time of day, time spent on the activity and number of attempts. Limit conditions for an activity include the associated data listed below.
SS processes may reference the limit conditions for any activity in the activity tree. Limit condition data need not be instantiated for each activity in the activity tree. Data for parent activities is used if data is not available for an activity.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Description</th>
<th>Value Space</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Limit Condition Attempt Control</td>
<td>Indicates that a limit condition on the number of attempts for the activity has been established for the activity. If the value is False, there is no constraint on how many times the activity may be attempted.</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>Limit Condition Attempt Limit</td>
<td>The maximum number of attempts for the activity. A zero value indicates the activity may not be accessed. The value is unreliable unless Limit Condition Attempt Control is True.</td>
<td>Non Negative Integer</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Limit Condition Activity Attempt WallClock Duration Limit</td>
<td>Indicates that a limit condition on the maximum time duration that a learner is permitted to spend on any single attempt on the activity has been established for the activity. If the value is False, there is no constraint on how long the learner may spend on any attempt.</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td>4</td>
<td>Limit Condition Activity Attempt WallClock Duration Limit</td>
<td>The maximum time duration that a learner is permitted to spend on any single attempt on the activity. A zero value indicates the activity may not be accessed. The value is unreliable unless Limit Condition Activity Attempt WallClock Duration Control is True.</td>
<td>Duration</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Limit Condition Activity Attempt Interaction Duration Control</td>
<td>Indicates that a limit condition on the maximum interaction time duration that a learner is permitted to spend on any single attempt on the activity has been established for the activity. If the value is False, there is no constraint on how long the learner may spend on the attempt.</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td>6</td>
<td>Limit Condition Activity Attempt Interaction Duration Limit</td>
<td>The maximum interaction time duration that a learner is permitted to spend experiencing a single attempt on the activity. The limit applies to only the time the learner is actually interacting with the activity and does not apply when the activity is suspended (i.e., when the activity is not being experienced or is inactive). A zero value indicates the activity may not be accessed. The value is unreliable unless Limit Condition Activity Attempt Interaction Duration Control is True.</td>
<td>Duration</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Limit Condition Activity WallClock Duration Control</td>
<td>Indicates that a limit condition on the maximum time duration that a learner is permitted to spend on the activity has been established for the activity. If the value is false, there is no constraint on how long the learner may spend on the activity.</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td>8</td>
<td>Limit Condition Activity WallClock Duration Limit</td>
<td>The maximum wall clock time duration that a learner is permitted to spend on all attempts at the activity. A zero value indicates the activity may not be accessed. The value is unreliable unless Limit Condition Activity WallClock Duration Control is True.</td>
<td>Duration</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Limit Condition Activity Interaction Duration Control</td>
<td>Indicates that a limit condition on the maximum interaction time duration that a learner is permitted to spend on the activity has been established for the activity. If the value is false, there is no constraint on how long the learner may spend on the activity.</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td>10</td>
<td>Limit Condition Activity Interaction</td>
<td>The maximum interaction time duration that a learner is permitted to spend on all attempts in experiencing the activity.</td>
<td>Duration</td>
<td>0</td>
</tr>
</tbody>
</table>
SM.1.4 Sequencing Thread Descriptions

Sequencing thread descriptions (the set of attributes shown below) define collections of activities that can be sequenced and delivered in parallel. Sequencing thread descriptions for an activity include the associated data listed below.

SS processes may reference the sequencing thread description for any activity in the activity tree. Sequencing thread descriptions need not be instantiated for each activity in the activity tree. Default data is used if the data is not instantiated for the activity.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Description</th>
<th>Value Space</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Thread ID</td>
<td>An ID number assigned to the activity. All activities with the same Thread ID are part of the same sequencing thread. The default value is the main sequencing thread.</td>
<td>Positive Integer</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Thread Label</td>
<td>An arbitrary label assigned to the thread.</td>
<td>String</td>
<td>“main”</td>
</tr>
</tbody>
</table>

SM.1.5 Rollup Rule Descriptions

Rollup rule descriptions (the set of attributes shown below) specify the details of individual rule-based rollup behaviors for an activity. Rollup rules describe how values of the child activities influence the Objective and Activity Progress data for an activity. Rollup rule description information for an activity includes the associated data listed below.

SS processes may reference the rollup rules for any activity in the activity tree. Rollup rules are instantiated only where needed. Default data is used if the data is not instantiated for the activity.

The data attributes describe one rule. Each activity may have an unlimited number of unordered rollup rules.

The general format of a rule can be expressed informally as: If child-activity set condition Then action.

- There is one child activity set that determines the child activities that participate in the condition.
- There is one condition that references an item in the tracking model for the child activity or its associated objective.
- There is one action that may result if the rule conditions evaluate to True.
Actions result in setting of activity progress data or objective data for the activity.

The Rollup Conditions and Rollup Actions are tokens in a vocabulary. The tokens have no semantics or meanings themselves. The description of the condition or action is the complete definition of the required behavior.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Description</th>
<th>Value Space</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Rollup Minimum Count</td>
<td>The number of children activities associated with a Rollup Child Set attribute value of At Least Count.</td>
<td>Integer</td>
<td>0</td>
</tr>
<tr>
<td>1.2</td>
<td>Rollup Minimum Percent</td>
<td>The percentage of children activities associated with a Rollup Child Set attribute value of At Least Percent.</td>
<td>Real [0..100]</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Rollup Condition</td>
<td>A condition element for the rule.</td>
<td>Vocabulary</td>
<td>Never</td>
</tr>
</tbody>
</table>

- **All** – The rollup rule condition evaluates to True if and only if all of the children have the value specified by the Rollup Condition.
- **Any** – The rollup rule condition evaluates to True if any of the children have the value specified by the Rollup Condition.
- **None** – The rollup rule condition evaluates to True if none of the children have the value specified by the Rollup Condition.
- **At Least Count** – The rollup rule condition evaluates to True if at least the number of children specified by the Rollup Minimum Count attribute have the value specified by the Rollup Condition.
- **At Least Percent** – The rollup rule condition evaluates to True if at least the percentage of children specified in the Rollup Minimum Percent attribute have the value specified by the Rollup Condition.

- **Satisfied** – evaluates to True if the Objective value of Objective Satisfied Status for the objective associated with the child activity (indicated by its Objective ID) is True.
- **Not Satisfied** – evaluates to True if the Objective value of Objective Satisfied Status for the objective associated with the child activity (indicated by its Objective ID) is False.
- **Objective Status Known** – evaluates to True if the Objective value of Objective Data Status for the objective associated with the child activity (indicated by its Objective ID) is True.
- **Completed** – evaluates to True if the Activity Progress value of Activity Completion Amount for the child activity is 1.0 (i.e., the child activity is complete).
- **Incomplete** – evaluates to True if the Activity Progress value of Activity Completion Amount for the child activity is not 1.0 (i.e., the child activity is incomplete).
incomplete).

- Activity Progress Known – evaluates to True if the Activity Progress value of Activity Completion Status for the child activity is True.
- Attempted – evaluates to True if the Activity Progress value of Activity Attempt Count for the child activity is positive (i.e., the child activity has been attempted).
- Attempt Limit Exceeded – evaluates to True if the Activity Progress value of Activity Attempt Count for the child activity is equal to or greater than the Limit Condition value of Limit Condition Attempt Limit for the child activity.
- Time Limit Exceeded – evaluates to True if any of the Activity Progress duration values for the child activity (Activity WallClock Duration, Activity Interaction Duration, Activity Attempt WallClock Duration, Activity Attempt Interaction Duration) exceed the corresponding duration Limit Condition values for the child activity (Activity WallClock Duration Limit, Activity Interaction Duration Limit, Activity Attempt WallClock Duration Limit, Activity Attempt Interaction Duration Limit).
- Outside Available Time Range – evaluates to True if the current time is before or after the corresponding time Limit Conditions for the child activity (Begin Time Limit, End Time Limit).
- Never – always evaluates to False.

<table>
<thead>
<tr>
<th>Rollup Action</th>
<th>The desired rollup behavior if the rule evaluates to True.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satisfied</td>
<td>The Objective value of Objective Data Status for the objective associated with the activity (indicated by Objective ID) is set to True. The Objective value of Objective Satisfied Status for the objective associated with the activity (indicated by Objective ID) is set to True.</td>
</tr>
<tr>
<td>Not Satisfied</td>
<td>The Objective value of Objective Data Status for the objective associated with the activity (indicated by Objective ID) is set to True. The Objective value of Objective Satisfied Status for the objective associated with the activity (indicated by Objective ID) is set to False.</td>
</tr>
<tr>
<td>Completed</td>
<td>The Activity Progress value of Activity Completion Status for the activity is set to True. The Activity Progress value of Activity Completion Amount for the activity is set to 1.0.</td>
</tr>
<tr>
<td>Incomplete</td>
<td>The Activity Progress value of Activity Completion Status for the activity is set to True. The Activity Progress value of Activity Completion Amount for the activity is set to a value not equal to 1.0.</td>
</tr>
</tbody>
</table>

**SM.1.6 Objective Description**

The objective description (the set of attributes shown below) defines the learning objective associated with an activity.
Each activity has one associated learning objective. The objective may be a unique objective for the activity or it may be some objective that is shared with other activities, enabling results to be shared among activities. The tracking model (see Tracking Model Document TM) defines a set of data that records the satisfaction status (e.g., passed/failed) and measure (e.g., score) for the objective. The meaning of a learning objective is not defined in this model; the objective is identified only by its ID.

SS processes may reference the objective information for any activity. The reference to the activity uses the objective ID to locate the actual objective data. Each activity must be assigned an objective ID.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Description</th>
<th>Value Space</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Objective ID</td>
<td>The identifier of the objective associated with the activity. The ID is a link to the corresponding objective information. The ID may be local to the activity, local to the activity tree or global. The ID is resolved to the most global scope possible to obtain the data. If the ID cannot be resolved, a local objective (and thus local objective information) is instantiated for the activity. Resolution of the scope of the ID is not specified in this model. Establishment of global registry of objective data is not specified in this model.</td>
<td>GUID</td>
<td>None – Value is Required, i.e., a unique local ID should be generated.</td>
</tr>
<tr>
<td>2</td>
<td>Objective Satisfied by Measure</td>
<td>Indicates that the Objective Minimum Satisfied Normalized Measure is to be used in place of any other method to determine if the objective associated with the activity has been satisfied.</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td>3</td>
<td>Objective Minimum Satisfied Normalized Measure</td>
<td>The minimum satisfaction measure for the objective, normalized between 0..1 (inclusive). If the Objective value of Objective Normalized Measure exceeds this value, the Objective value of Objective Data Status is set to True and the Objective value of Objective Satisfied Status is set to True. The value is unreliable unless Objective Satisfied by Measure is True.</td>
<td>Real [0..1]</td>
<td>1.0</td>
</tr>
</tbody>
</table>

**SM.1.7 Rollup Controls**

Rollup controls (the set of attributes shown below) include descriptions of the types of rollup behaviors specified for an activity. Rollup controls for an activity include the associated data listed below.

SS processes may reference the rollup control data for any activity in the activity tree. Rollup control data need not be instantiated for each activity in the activity tree. Data for parent activities is used if data is not available for an activity.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Description</th>
<th>Value Space</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rollup Objective Satisfied</td>
<td>Indicates that the Objective value of Objective Satisfied Status for the objective associated with the activity (indicated by Objective ID) is included (True or False) in the rollup for the parent activity.</td>
<td>Boolean</td>
<td>True</td>
</tr>
<tr>
<td>2</td>
<td>Rollup Objective Measure Weight</td>
<td>A weighting factor applied to the Objective value of Objective Normalized Measure for the objective associated with the activity (indicated by Objective ID) in the rollup for the parent activity.</td>
<td>Real [0..1]</td>
<td>1.0</td>
</tr>
</tbody>
</table>

© Copyright 2002, Carnegie Mellon University, All Rights Reserved.
SM.1.8 Delivery Controls

Delivery controls (the set of attributes shown below) describe actions and controls used when an activity is delivered, i.e., Objective and Activity Progress Data are recorded when the activity is delivered. Delivery controls for an activity include the associated data listed below.

SS processes may reference the delivery control data for any activity in the activity tree. Delivery data need not be instantiated for each activity in the activity tree. Data for parent activities is used if data is not available for an activity.

The **Timeout Actions** are tokens in a vocabulary. The tokens have no semantics or meanings themselves. The description of the condition or action is the complete definition of the required behavior.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Description</th>
<th>Value Space</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tracked</td>
<td>Indicates that Objective and Activity Progress data for the activity attempt should be recorded (True or False) and the data will contribute to the rollup for the parent activity. How the data is tracked and recorded is not specified.</td>
<td>Boolean</td>
<td>True</td>
</tr>
<tr>
<td>2</td>
<td>Completion Set by Content</td>
<td>Indicates that the Activity Progress completion data for the activity in the Tracking Model will be set by the content object (True or False). A False value indicates that default rules will be used to set progress data. For a True value, how, if or when the completion data is set is not specified.</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td>3</td>
<td>Objective Set by Content</td>
<td>Indicates that the Objective data for the objective associated with the Tracking Model will be set by the content object (True or False). A False value indicates that default rules will be used to set objective data. For a True value, how, if or when the objective data is set is not specified.</td>
<td>Boolean</td>
<td>False</td>
</tr>
</tbody>
</table>
| 4   | Timeout Action                         | Indicates the desired action when a duration limit is exceeded and the activity is being experienced:  
  • **Continue** – The activity continues.  
  • **Continue With Message** – The activity continues. The delivery system issues a message to inform the learner that a duration limit condition has been exceeded.  
  • **Terminate** – The attempt on the activity terminates.  
  • **Terminate With Message** – The attempt on the activity terminates. The delivery system issues a message to inform the learner that a duration limit condition has been exceeded.  
  Mechanisms to terminate the activity or cause the message to be displayed are not defined in the data model and may not be supported by some delivery systems. | Vocabulary   | Continue      |
This model retains equivalent functionality and intent with that described in the IMS SS Public Draft. Descriptions and terminology have been simplified. Detailed behavior descriptions have been removed and are defined in only the behavior model (see Behavior Models Documents).

Items that are logically related are split into several top-level groups. Each group describes a set of related attributes for an activity.

This model eliminates data items that are not related to sequencing or are otherwise incomplete. These data items may be important for an implementation but do not belong in the SS specification.

- **Data Mapping**: This is an application profile–specific extension to SS. Data Mapping describes how one particular type of content relates its private data values to the data attributes of the SS tracking model. Data Mapping does not define any sequencing behavior. Data Mapping is not used in any sequencing decisions.

- **Persistent State**: The Persistent State attribute controls how and when a content object’s state data is persisted. Persistent State does not define any sequencing behavior. Persistent State is not used in any sequencing decisions.
  - Dropping Persistent State has no effect, i.e., the same sequence is selected. The persistent state information is important for knowing how the content behaves, but is important only within the scope of an individual content object.
  - The model to define persistence is complex. Persistence is defined as an attribute of a resource associated with an activity, not of the activity—the same resource used in different activities share the persisted data. For example, 4 activities that share the same resource share the same state. Using the data item, 2 of the activities could be marked as not having persistent state, and thus not sharing data. To have the other two activities not have shared state but to still have persistent data, a dummy resource for each has to be created which then references the actual resource.
  - Sharing of data via persistent state creates the opportunity for back door communications between activities and essentially morphs the activity tree into a graph.

- **Communicative**: Use of this data item does not change how content is sequenced. Content being communicative is not used in any sequencing decisions. Sequencing relies only on the values of state. Sequencing does not know how, when or where the values of state are set or changed. The sequencing or delivery processes do not assume that content communications is reliable and thus cannot differentiate between a content object that does not communicate and one that fails to communicate properly. If it is important to know that a content object uses a particular communication model (e.g., the CMI API), this should be represented in the metadata for the content object.

- **Score Required**: Use of this data item does not change how content is sequenced. Sequencing relies only on the values of state. It does not know how, when or where the values of state are set or changed. The sequencing process cannot differentiate between an activity for which a score is reported and one for which the score is not reported, and also cannot differentiate between an activity that does not report a score and a communications failure in reporting the score. In addition, the specified default required score of 0 is the default score that the tracking model will return. The default adds no new data and does not change an existing behavior. Rollup is triggered when specified in the behavior model and uses the scores available at that time. Availability of a score does not change the rollup process and the defaults result in a consistent behavior, independent of the existence of this data item.

- **Reinitialize**: Use of this data item does not make sense. The default behavior is to delete all status information for all subactivities whenever an attempt on a parent activity is ready to be delivered. Essentially, in choice mode, any selection deletes all tracking data for all children. If the element were to remain, the default must be not to reinitialize. The description does not explain how reinitialization impacts shared competency references or objective data. As described, any attempt would destroy the competency records, including any globally shared record.

- **Randomize**: Randomize is a description of a restructuring of the activity tree. The effect of randomize is to change the tree. SS makes no requirements on how or when the tree is created or instantiated. The effect of randomize can be accomplished by dynamically creating a specific sequencing tree that orders the activities in the proper way so that the SS process produces the desired behavior. Randomize as an attribute of an activity tree is not required for a sequencing process to produce the specified behavior.
The following data items have been changed.

**Sequencing Control Mode:**
- The default for *Sequencing Control Forward Only* is changed to False. The default corresponds to the current Content Package that does not constrain sequencing.

**Sequencing Rule Definitions:**
- Objective and Activity status value conditions are changed to correspond to the tracking model.
- An OK action is added since all rules require an action, and an appropriate default is needed.
- Time limit checks modified to apply to all four cases (activity and attempt duration, activity and attempt interaction time).
- Why does the Skip condition not apply in choice mode? The logical combinations are very confusing and seem to eliminate some basic behaviors. In choice mode, there is no automatic flow if the choice is not available, so Skip simply results in a condition where there is nothing to deliver. The end result is the same if the logic is simplified and actions are independent. The simplified logic is used here.
- Additional actions to re-enable disabled or hidden activities added for completeness.
- The exit behaviors were changed from sequencing requests to exit requests. As defined, a sequencing request could result in another sequencing request. Such recursive processing is overly complex and not needed to obtain the desired behavior.

**Limit Conditions:**
- Control values added for all items to eliminate multi value logic.
- The value space for Attempt Limits changed to nonnegative, permitting a zero limit that could be used to disable access to the activity.
- Limit condition changed so that there can be limits on both the duration and interaction time for both an attempt and an activity (4 cases).
- Continue option added to the Timeout Action. The four possible cases are now all represented: Continue or Terminate, with or without message.
- The Continue option is the default Timeout Action.
- Timeout Action is not a limit condition, but a delivery control.

**Sequencing Threads:**
- Changed the single attribute into a type attribute (e.g., used for display) and an ID used for control.

**Rollup Rule Descriptions:**
- Converted to the same “if condition then action” format as sequencing rule definitions. This is a data representation change only, not a change in any defined behavior.

**Objective Description:**
- Added the Satisfied by Measure attribute to control if measure-based (e.g., score) rollup is used.

**Rollup Controls:**
- No changes.

**Delivery Controls:**
- Deliverymode is converted to a delivery control (*Tracked*) that is a binary value. Browse and Review behaviors are identical from the sequencing perspective. Deliverymode as a CMI content attribute is part of the CMI model, not SS.

The following data items have been added:

**Progress Set by Content / Objective Set by Content:**
• The attributes are used to control if the sequencing processor sets activity completion or objective data, or if it relies on the content object to set the data. SS makes no assumptions about the actual behavior of the content.

References to SS behaviors, application profile–specific behaviors and other descriptions not pertaining to just the sequencing description have been eliminated. This part only describes the data model, not how it is used by the sequencing process.

**SM.3 Open Issues**

Some of these issues may be resolved when the behaviors are properly defined and fully described.

The order of the element groups is not final. It could be useful to specify rendering requests, e.g., timeout action, as a separate element group. [SS.SM.01]

A section that defines the elements of the value spaces is needed. [SS.SM.02]

In all cases, the inheritance of values from the parent versus the use of default values from the data model needs to be verified. [SS.SM.03]

Do sequencing controls apply to sequencing requests or navigation requests? The name should match the type of request. Sequencing has been used throughout. [SS.SM.04]

Is the Forward Only sequencing request applicable to Flow requests, Previous requests, or both? [SS.SM.05]

Sequencing control modes are defined for an activity cluster. All other data is defined for an activity. It would simplify the behavioral model to describe all data consistently. There is no requirement that the sequencing control modes only be defined for the parent of the cluster and used for the children. The data can be fully instantiated for the children. Defining the data at the aggregate level may be useful for authoring, but this need not determine how the data model is defined. [SS.SM.06]

The sequencing rules include conditions to test if activities (but not attempts) are completed. The rules include conditions to test if activities or attempts exceed time limits. Should the inconsistency remain? Should rules on attempt time limits be dropped, or should rules on attempt completion be added? [SS.SM.07]

Forward only and other complex rule actions are difficult to understand. The intermixing of data and behavior complicates the presentation. Fundamental behaviors may require additional data items in either rule conditions or actions. Thus the rule descriptions are incomplete until the behaviors have been properly described. [SS.SM.08]

Does the sequencing rules data model need conditions that are the logical negative of each other? An authoring language may benefit from even more logical conditions than those given, but the sequencing behavior can be described in terms of the simpler set of equivalent conditions. The logical opposite satisfied and not satisfied conditions could be reduced to one set. [SS.SM.09]

The disable sequencing rule action implies that there is some record of the status of the activity, and by implication, the need for an alternative action to re-enable the rule. The action appears to be a “do not deliver” precondition for the activity and probably requires a more logical name. [SS.SM.10]

Do limit conditions remain in whole or part? In particular, the start/end time model provides only a subset of values and is hard to extend as defined. [SS.SM.11]

If there is a duration limit on the aggregation, does this apply to the individual activities within the aggregation? Does the parent govern if the data is missing? It seems that it is hard enough to track the time on the activity and the attempt, let alone monitor the real time rollup of time across all the parents. As specified, the model seems to be inconsistent in that date restrictions do inherit values. [SS.SM.12]
The clock (local, server, UTC) used to evaluate time limits is not specified. [SS.SM.13]

Limit condition exceedance actions are not all defined. Timeout behaviors are only one action and described as part of a delivery action. The deliver action itself is not part of SS. Timeout action should be removed. A collection of other actions are needed to determine when a limit is exceeded. [SS.SM.14]

How does rollup score weight combine with rollup rules? Both provide a way to combine scores. Which governs? [SS.SM.15]

Do the rollup controls eliminate potential members in the child set for rollup rules, or do they have some other purpose? [SS.SM.16]

It is unclear how delivery control (delivery mode) interacts with rollup. If data is not tracked, are the rollup results the same if rollup is not done (i.e., tracking false does not track or rollup)? The two approaches may be in conflict. Since it is impossible to insure that progress data is not updated, should rollup be disabled when tracking is off? [SS.SM.17]

Randomize illustrates a core problem with the sequencing definition model. The behavior model needs a precise set of data attributes for each activity. For many elements, the sequencing definition model describes the data that is instantiated and used for each activity. For other elements, e.g., randomize, the model defines an overall behavior. Similarly, the sequencing control modes do not define controls for an activity but for a cluster. The sequencing definition model attempts to provide both the precise data model used for behaviors and a model that is convenient for a content author to describe an activity sequence. [SS.SM.18]

This intermixing of concepts complicates SS. Is the definition model that which is used by the SS process, or is the definition model that which is used to describe content to the outside world, i.e., an authoring model? The data model is not the appropriate way to express a control or definitional language – the language could simply specify behaviors and not the underlying data model used to represent these. The attempt to use the binding of the data model as the authoring language further clouds the issue of what the purpose of the data model is. The definition model tries to do both, but it does not fully detail either approach. [SS.SM.19]

Throughout this document, the focus is on the activity description, not a content authoring model. Authoring systems may use other representations and map these to the precise descriptions used to drive the sequencing process. [SS.SM.20]

The relationship among sequencing, navigation and rollup, particularly through the use of exit sequencing rules, is not clear. It appears that an exit sequencing request may not be a request in the same sense as a sequencing request that walks the activity tree, but rather is simply a request to terminate some set of activities prior to or in conjunction with rollup, and is usually triggered by a navigation event. [SS.SM.21]

If Objective Normalized Measure is normalized from –1..1 then Objective Minimum Satisfied Normalized Measure should also be normalized from –1..1. [SS.SM.22]

Changing Rollup Objective Measure Weight to be normalized form –1..1 may be useful. Generalized positive and negative weights are more meaningful that just positive weights and may be useful in describing learning activities where objectives are used to control satisfaction or dissatisfaction. [SS.SM.23]

Threads are not fully described in behaviors and could be dropped. [SS.SM.24]

There is no mechanism to insure that the Thread ID and Thread Label are consistent. Thread descriptions should be global to the activity tree, not local to activities. [SS.SM.25]

Are rules (sequencing and rollup) unordered as defined? [SS.SM.26]

Are conditions unordered as defined? [SS.SM.27]
Should there be multiple objectives for an activity. The complexity comes in specifying in the sequencing and rollup rules which of the objectives associated with the activity are used in rule evaluation (e.g., a particular objective only, a set, all). [SS.SM.28]

SS requires a complete binding description. The binding description should be human readable, natural language, independent of the XML schema. The binding description must include:

- How one or more data model elements are mapped to one or more items in the binding.
- The names of the data model element in the binding that correspond to the names in the data model.
- The XML schema data type for the data items.
- Constraints on the data type.
- Vocabulary encodings.
- Constraints on an implementation, e.g., minimum string sizes, minimum sizes of collections that a conforming system must support. [SS.SM.29]

**SM.4 Implementation Recommendations**

Values of *Completion* and *Measure* are normalized between 0..1. Testing for the exact value of 1.0 is unreliable. Any value > 0.99999 should be considered equal to 1.0 in evaluating rule conditions (based on the required precision of 5 decimal digits).

Normalized data values are described as value space real. A decimal data type should be considered as an alternative to a float data type for the implementation to maintain the required precision.

Rule processors should use short circuit evaluation techniques when determining if the rule conditions are true. For example, assume an *And* rule contains a condition on both *Objective Data Status* and *Objective Satisfied Status*. If the *Objective Data Status* evaluates to False, the condition on the the value of *Objective Satisfied Status* should not be evaluated (in this case, the *Objective Satisfied Status* value is unreliable). The rule processor may reorder conditions to enable short circuit evaluation. However, short circuit evaluation is not required, as the default values insure that the same sequencing behaviors will result whether short circuit evaluation is used or not.

An implementation may use a general rule representation and encode the specific sequencing rules as instances of the more general representation.

**SM.5 Bindings**

The data elements are defined using the names used in the SS behavior models. The XML schema used to exchange sequencing definitions (see Binding Document) uses a different set of names (unchanged from the Public Draft). The binding document describes the mapping from the names used in the data model to those used in the schema, and includes the complete hierarchical structure of data items needed to express the association of data items with activities and elements of the activity tree.
TM. Tracking Model (Just Tracking V1.0)

The SS process uses information about the results of a learner’s interactions with activities, and the learner’s record for objectives (e.g., completion, measure) to control the selection and sequencing of other activities. The sequencing behaviors are defined in terms of a limited set of specific data attributes that describe the results of the learner’s interactions. The defined set of attributes used by SS is called the “tracking model”.

The tracking model consists of:
- An information model, defined in two parts:
  - Objective Information – information about the results of the learner’s interactions related to an objective.
  - Activity Progress Information – information about a learner’s attempt at an activity.
- Behaviors – requirements on instantiation and use of the tracking information model.

TM.1 Tracking Information Model

The tracking information model describes the data used by a system that delivers sequenced activities. How this information is encoded, stored, represented or bound is outside the scope of this specification. The tracking information model only describes a set of related data items and internal constraints on the values of those items.

An implementation must be capable of representing the range of values described. There are no additional requirements on implementing the information model.

The tracking model behaviors description states requirements for the instantiation and use of the information model.

TM.1.1 Objective Information

Objective information includes results of the learner’s interactions related to an objective. Objective information (the set of attributes shown below) is associated with the objective itself, independent of how the objective is associated with any activity. Tracking information for an objective includes the associated data (i.e., the objective information) listed below. How the tracking information is associated with an objective is not defined in this model.

SS descriptions may reference objective data for any objective associated with any activity in the activity tree. Objective data should be instantiated for objectives referenced in the activity tree for each learner.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Description</th>
<th>Value Space</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Objective Data Status</td>
<td>Indicates if the other values of the objective information for the objective are valid.</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>Objective Satisfied Status</td>
<td>Indicates the objective is satisfied (True or False).</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The value is unreliable unless Objective Data Status is True.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The determination or meaning of satisfied or not is not defined in this model.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Objective Measure Status</td>
<td>Indicates the objective has a measure value (True or False).</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The value is unreliable unless Objective Data Status is True.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Objective Normalized Measure</td>
<td>The measure (e.g., score) for the objective, normalized between 0..1 (inclusive). The value is</td>
<td>Real [0..1]</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>unreliable unless Objective Data Status and Objective Measure Status are True. The mechanism</td>
<td>Precision of at least 5 significant decimal digits</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>to normalize a measure is not defined in this model.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TM.1.2 Activity Progress Information

Result information (the set of attributes shown below) about a learner’s progress and attempts at an activity. This information describes cumulative progress for an individual activity and data for a single attempt at an activity. Tracking information for an activity and attempt includes the associated data listed below.

SS descriptions may reference activity and attempt progress data for any activity in the activity tree. Activity and attempt progress data should be instantiated for each activity in the activity tree for each learner. How the activity progress information is associated with an activity is not specified.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Description</th>
<th>Value Space</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Activity Completion Status</td>
<td>Indicates the activity completion data is meaningful for the activity (True or False).</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>Activity Completion Amount</td>
<td>The measure of the completion of the activity, normalized between 0..1 (inclusive) where 1 means the activity is complete and any lesser value means the activity is not complete. The value is unreliable unless Activity Completion Status is True. The mechanism to define the completion amount is not defined in the information model.</td>
<td>Real [0..1] Precision of at least 5 significant decimal digits</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>Activity WallClock Duration</td>
<td>The cumulative duration of all attempts on the activity, i.e., the time from the initial start of the activity to the end of the activity. The value is unreliable unless Activity Completion Status is True. The mechanism for determining the duration is not defined in this model.</td>
<td>Duration</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Activity Interaction Duration</td>
<td>The cumulative interaction duration of all attempts on the activity, i.e., the time from the initial start of the activity to the end of the activity, not including any time elapsed while the activity is suspended (i.e., when the activity is not being experienced or is inactive). The value is unreliable unless Activity Completion Status is True. The mechanism for determining the duration or the suspend time is not defined in this model.</td>
<td>Duration</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Activity Attempt Count</td>
<td>The number of attempts on the activity. The count includes the current attempt, i.e., 0 means the activity was not attempted and 1 or greater means it either is in progress or completed. The mechanism and timing of when the Activity Attempt Count is incremented is not defined in this model.</td>
<td>Non-negative integer</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>Activity Attempt Completion Status</td>
<td>Indicates the activity attempt completion data (True or False) is meaningful for the activity attempt. The value is unreliable unless Activity Attempt Count is greater than (&gt; ) 0.</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td>7</td>
<td>Activity Attempt Completion Amount</td>
<td>The measure of the completion of the attempt on the activity, normalized between 0..1 (inclusive) where 1 means the activity attempt is complete and any lesser value means the activity attempt is not complete. The value is unreliable unless Activity Attempt Completion Status is True. The mechanism to define the completion amount is not defined in this model.</td>
<td>Real [0..1] Precision of at least 5 significant decimal digits</td>
<td>0.0</td>
</tr>
</tbody>
</table>
### TM.2 Tracking Model Behaviors

The objective data applies to an objective. It should be instantiated for each objective for each learner.

The activity progress data is for an activity. It should be instantiated for each activity for each learner.

The SS process accesses the activity progress data only for the most recent attempt at an activity by a learner. If the learner is interacting with the activity, for the purposes of the tracking model, the most recent attempt is the current interaction or current attempt. Otherwise, the most recent attempt is the last fully completed interaction with the activity (if there was one), i.e., the last completed attempt.

There is no requirement that the activity progress data be maintained for any attempt other than the most recent attempt. SS makes no requirement to maintain prior history for objective information or overall activity information.

There are no requirements on how or when the values of the tracking data are set and updated for the learner. SS relies only on the “current” values in the information model. Generally, learner interactions with an activity will result in appropriate changes to the tracking data, but these mechanics are not specified.

The rollup process (see Rollup Behavior Document RB) uses the tracking data for one activity to determine tracking data for other activities and their associated objectives. How the rollup process uses the tracking data is not specified here but is detailed in the description of the behavior of the rollup process.

The sequencing and delivery processes (see Sequencing Behavior Document SB, Delivery Behavior Document DB) use the tracking data for elements of the activity tree to determine the activities that are sequenced and delivered to the learner. How the sequencing and delivery processes use the tracking data are not specified here but are detailed in the descriptions of the behavior of these processes.

There is no requirement that the value for any specific tracking data item exist. There is a defined default or initial value for each attribute. The default value is supplied upon reference if the data value does not exist.

The mechanisms used to record and set the tracking data are not specified as part of the information model.

### TM.3 Notes/Changes from IMS SS Public Draft

This model is identical in functionality and intent with that described in the IMS SS Public Draft. Descriptions and terminology have been simplified.
References to SS behaviors, application profile–specific behaviors and other descriptions not pertaining to just the tracking model have been eliminated. This part just describes the tracking model itself, not how it is used by the sequencing process.

The objective and activity progress information have been split into two separate parts in the information model. The information model is based on the simplification that all objectives, either for an activity or for a shared competency, are the same – just an objective. There is data for an objective, but how the objective is associated with an activity or competency definition is out of scope for the tracking model itself. Similarly, there is data for an activity attempt, independent of the objectives associated with the activity. Thus the data items are split along these lines.

The terminology of passed and score for an objective have been generalized as a satisfaction state and a satisfaction measure. This change clarify that objectives have no particular pedagogical meaning but are simply a state and measure value that can be shared across activities to control sequencing behaviors.

The tracking information model splits some of the vocabularies into sets of binary values: e.g., the MasterStatus.PassFail vocabulary of pass, fail, or unknown is split into Objective Status (Y/N), and Satisfied (Y/N). N-value logic (e.g., pass/fail/unknown) is confusing and difficult to implement and describe. Splitting the values makes the rest of the SS behaviors easier to describe. There is no requirement that an implementation use separate values, just that it provides the prescribed behaviors, i.e., an implementation may encode the Objective Status and Satisfied attributes into a single data item.

All descriptions of behaviors that the SS process imposes on the values of items in tracking models are deleted. The tracking model is only the set of data, not a specific application profile use of the data.

**TM.4 Open Issues**

Is there a better or more understandable name than “tracking model”? The model is actually just a subset of full tracking data. [SS.TM.01]

The duration times are not defined when an activity or attempt is in progress. It is unclear how sequencing behaviors should interpret the data for an activity or attempt that is in progress. [SS.TM.02]

Should there be an additional attribute to designate if times are actually tracked (a boolean)? The durations would only be valid if the value is true – but this is a change from the current model, which says there must be timing data. Currently there is no mechanism to differentiate between untracked and a time value of zero. [SS.TM.03]

The duration items are not required if limit conditions are dropped. [SS.TM.04]

It may be useful to define Activity Completed (boolean) and Attempt Completed (boolean). Testing the boolean is easier and more reliable than testing that Activity Completion is close to 1.0 (within 5 digits). This would permit the SS behaviors to be described in terms of explicit state values and leave it to the implementation to determine how the normalized completion value maps to the boolean state. [SS.TM.05]

It may be useful to have Objective Normalized Measure be normalized from −1..1. While a negative score may not be meaningful, the generalization as positive and negative measures are more meaningful and may be useful in describing learning activities where objectives are used to control satisfaction or dissatisfaction. [SS.TM.06]

**TM.5 Implementation Recommendations**

The implementation must ensure that data is updated in the proper sequence. For example, the value of Objective Data Status should not be set to True until there is a value of Objective Satisfied Status. Reversing the order of assignment of these two values could result in a situation where an accessor uses a True value of Objective Data Status to determine if it can access Objective Satisfied Status, and then accesses Objective Satisfied Status before the second value assignment is complete. Implementing proper transactional controls are left to the implementation.
Values of Completion and Measure are normalized between 0..1. Testing for the exact value of 1.0 is unreliable. Any value > 0.99999 should be considered equal to 1.0 (based on the required precision of 5 decimal digits).

Normalized data values are described as value space real. A decimal data type should be considered as an alternative to a float data type for the implementation to maintain the required precision.

TM.6 Bindings

The tracking model is used only to control sequencing. There is no binding or other exchange mechanism. A binding is not required.

The SS behavior model describes how the tracking data is used. The SS binding provides the mapping of the tracking data items to those used to describe sequencing behaviors.
AM. Activity State Model (Just Activity State V0.4)

The SS process uses information about the state or status of the learner’s interactions with activities. These attributes are used to control the overall sequencing process, but are not referenced in any specific sequencing definition or sequencing rule. The defined set of state attributes used by SS is called the “state model”.

The state model consists of:
- An information model, defined in two parts:
  - Activity State Information – information that describes the state of the current attempt at an activity.
  - Global State Information – information that describes the overall state of sequencing.
- Behaviors – requirements on instantiation and use of the state information model.

AM.1 State Information Model

The state information model describes the data used by a system that delivers sequenced activities. How this information is encoded, stored, represented or bound is outside the scope of this specification. The state information model only describes a set of related data items and internal constraints on the values of those items.

An implementation must be capable of representing the range of values described. There are no additional requirements on implementing the information model.

The state model behaviors description gives requirements for the instantiation and use of the state information model.

AM.1.1 Activity State Information

Activity state information (the set of attributes shown below) describing a learner’s state or status for an activity.

SS descriptions and behaviors may reference state activity data for any activity in the activity tree. Activity state data should be instantiated for each activity in the activity tree for each learner. How the activity state information is associated with an activity is not specified.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Description</th>
<th>Value Space</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Activity In Progress</td>
<td>Indicates the activity is currently being experienced, i.e., has been delivered to the learner and has not been exited (True or False).</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>Activity Suspended</td>
<td>Indicates the activity is currently suspended (True or False).</td>
<td>Boolean</td>
<td>False</td>
</tr>
</tbody>
</table>

AM.1.2 Global State Information

Global state information (the set of attributes shown below) describing a learner’s state or status within the overall learning experience.

Global state information should be instantiated once for an activity tree for each learner. How the global state information is associated with an activity tree is not specified.
<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Description</th>
<th>Value Space</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Current Activity</td>
<td>Indicates the current activity in the activity tree being experienced by the learner.</td>
<td>Activity Designation</td>
<td>None</td>
</tr>
</tbody>
</table>

**AM.2 Activity State Model Behaviors**

The activity state data is for an activity. It should be instantiated for each activity for each learner.

The global state data is for the overall sequencing process. It should be instantiated for the entire sequencing process for each learner.

The SS process accesses the state data only for the most current attempt at an activity for a learner.

The navigation, exit, sequencing, rollup and delivery processes (see Navigation Behavior Document NB, Exit Behavior Document EB, Sequencing Behavior Document SB, Rollup Behavior Document RB, Delivery Behavior Document DB) use the activity state data for elements of the activity tree to determine the activities that are sequenced and delivered to the learner. How the sequencing and delivery processes use the activity state data are not specified here but are detailed in the descriptions of the behavior of these processes.

There is no requirement that the value for any activity state data item exist. There is a defined default or initial value for each attribute. The default value is supplied upon reference if the data value does not exist.

The mechanisms used to record and set the activity state data are not specified as part of the information model.

**AM.3 Notes/Changes from IMS SS Public Draft**

This model is identical in functionality and intent with that described in the IMS SS Public Draft. Descriptions and terminology have been simplified.

This is a new information model. The activity and global state data are implicit in the IMS SS Public Draft. This model makes the data explicit so it can be referenced and used in a consistent manner throughout the rest of SS.

**AM.4 Open Issues**

Is there a better or more understandable name than “state model”? [SS.AM.01]

Are there other key elements of a state model that should be defined? [SS.AM.02]

**AM.5 Implementation Recommendations**

The implementation must ensure that data is updated in the proper sequence. For example, the value of Activity Suspended should not be set to True unless the activity has been suspended in the delivery environment. Implementing proper transactional controls are left to the implementation.

**AM.6 Bindings**

The state model is used only to control sequencing. There is no binding or other exchange mechanism. A binding is not required.

The SS behavior models describes how the state data is used.
NB. Navigation Behavior Model (Just Navigation V0.5)

When a learner is interacting with a learning experience through a user interface, each learner request to move through the content or branch within the learning experience results in a “navigation event”, e.g., a click to move to the next activity. These events map to a set of “navigation requests”. In turn, each navigation request maps to an “exit request” and a “sequencing request” that will be used to both terminate the current activity and determine the next activity in the learning experience. There is a mapping or resolution process that converts a navigation request into the corresponding exit and sequencing requests. The process of evaluating the navigation request and identifying the appropriate exit and sequencing requests (or returning an error) is called the “navigation process”.

The navigation process makes no assumptions as to how or when a navigation request is generated. The navigation process makes no assumptions about user interfaces and rendering of navigation structures and controls, either within content objects or external to content objects. Mapping of a user interaction to the generation of an actual navigation event is outside of the scope of SS. The navigation behavior is defined only for the actual navigation request.

The navigation process must insure that navigation requests are valid. The navigation process makes no assumptions on how or when the identified exit and sequencing requests will be processed. In most cases, the exit and sequencing requests will be passed immediately to the exit and sequencing processes.

The overall sequencing process (see Overall Sequencing Process Document OP) relates the navigation process to the sequencing, delivery, exit, and rollup processes.

The navigation process does not require data from the sequencing definition model.

The navigation process does not use data from the tracking model.

The navigation process uses data from the activity state model.

The behavior of the navigation process is defined in terms of a single process:
- Navigation Request Process – processes the actual navigation request to identify the corresponding sequencing and exit requests.

NB.1 Navigation Requests

The navigation process responds to one of a set of different navigation requests. A navigation request defines a set of actions to be performed when the request is processed. The request names are tokens in a vocabulary. The names have no semantics or meanings themselves. The definition of the action is the complete definition of the required behavior.

<table>
<thead>
<tr>
<th>Navigation Request</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Issue a Start sequencing request.</td>
</tr>
<tr>
<td>Continue</td>
<td>Issue a Continue sequencing request.</td>
</tr>
<tr>
<td>Previous</td>
<td>Issue a Previous sequencing request.</td>
</tr>
<tr>
<td>Forward</td>
<td>Issue a sequencing request to traverse the “history-based activity record” forward in time. The corresponding sequencing request and associated behavior is not specified in this version of SS.</td>
</tr>
<tr>
<td>Backward</td>
<td>Issue a sequencing request to traverse the “history-based activity record” backward in time. The corresponding sequencing request and associated behavior is not specified in this version of SS.</td>
</tr>
</tbody>
</table>
### Choice

Issue a *Choice* sequencing request. The request is accompanied by the identification of the target activity.

### Exit

Issue an *Exit* sequencing request.

The current attempt on the currently delivered activity is terminated normally; the attempt is over. The termination and exit of the activity was not the result of any other external navigation event (e.g., *Continue, Previous, Choice*).

### Exit All

Issue an *Exit All* sequencing request.

### Suspend

Issue an *Exit* sequencing request.

Issue a *Suspend* exit request.

The current attempt on the currently delivered activity is terminated normally; the attempt is not over, and the activity is not complete. The activity may be resumed at some time in the future (resumption is not a new attempt). The SS processor must record sufficient state and tracking information so that the activity may be resumed in the future.

### Suspend All

Issue an *Exit All* sequencing request.

Issue a *Suspend All* exit request.

The current attempt on the currently delivered activity and all of its parents are terminated normally; the attempts are not over, and the activities are not complete. The activities may be resumed at some time in the future (resumption is not a new attempt). The SS processor must record sufficient state and tracking information so that the activities may be resumed in the future.

### Abandon

Issue an *Exit* sequencing request.

Issue an *Abandon* exit request.

The current attempt on the currently delivered activity is terminated abnormally and the activity is not complete. The activity attempt may not be resumed. There is no rollback of any tracking data.

### Abandon All

Issue an *Exit All* sequencing request.

Issue an *Abandon All* exit request.

The current attempt on the currently delivered activity and all of its parents are terminated abnormally and the activities are not complete. The activity attempts may not be resumed. There is no rollback of any tracking data.

---

**NB.2 Navigation Behavior**

The navigation behavior describes how a navigation processor interprets a navigation request to validate the navigation request and to identify the corresponding exit and sequencing requests.

An implementation must be capable of representing the processes described and have the implemented process exhibit the behavior described. There are no additional requirements on implementing the navigation behavior model.

**NB.2.1 Navigation Request Process**

The *Navigation Request Process* examines the navigation request and determines the corresponding exit and sequencing requests. The process implements the behaviors that describe the navigation requests.

The *Navigation Request Process* for a navigation request is specified by the following pseudo code. The pseudo code describes only the request processing and conversion to the exit and sequencing requests. How this process is
implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The Navigation Request Process only describes the expected behavior that an implementation will exhibit.

**Navigation Request Behavior**

1. **Case:** Navigation Request is *Start*
   1.1. **If** there is not an active attempt on a delivered activity **Then**
      1.1.1. Sequencing Request is *Start*
      1.1.2. Sequencing Request location is root of activity tree
      1.1.3. **Exit Navigation Request Process** – Navigation Request is valid
   1.2. **Else**
      1.2.1. **Exit Navigation Request Process** – Navigation Request is not valid

2. **Case:** Navigation Request is *Continue*
   2.1. **If** there is an active attempt on a delivered activity **Then**
      2.1.1. Exit Request is *Exit*
      2.1.2. Sequencing Request is *Continue*
      2.1.3. **Exit Navigation Request Process** – Navigation Request is valid
   2.2. **Else**
      2.2.1. **Exit Navigation Request Process** – Navigation Request is not valid

3. **Case:** Navigation Request is *Previous*
   3.1. **If** there is an active attempt on a delivered activity **Then**
      3.1.1. Exit Request is *Exit*
      3.1.2. Sequencing Request is *Previous*
      3.1.3. **Exit Navigation Request Process** – Navigation Request is valid
   3.2. **Else**
      3.2.1. **Exit Navigation Request Process** – Navigation Request is not valid

4. **Case:** Navigation Request is *Forward*
   4.1. Behavior is not defined
   4.2. **Exit Navigation Request Process** – Navigation Request is not defined

5. **Case:** Navigation Request is *Backward*
   5.1. Behavior is not defined
   5.2. **Exit Navigation Request Process** – Navigation Request is not defined

6. **Case:** Navigation Request is *Choice*
   6.1. **If** there is an active attempt on a delivered activity **Then**
      6.1.1. Exit Request is *Exit Target*
      6.1.2. Sequencing Request is *Choice*
      6.1.3. Sequencing Request location is as specified in the Navigation Request
      6.1.4. **Exit Navigation Request Process** – Navigation Request is valid
   6.2. **Else**
      6.2.1. **Exit Navigation Request Process** – Navigation Request is not valid

7. **Case:** Navigation Request is *Exit*
   7.1. Exit Request is *Exit*
   7.2. Sequencing Request is *Exit*
   7.3. **Exit Navigation Request Process** – Navigation Request is valid

8. **Case:** Navigation Request is *Exit All*
   8.1. Exit Request is *Exit All*
   8.2. Sequencing Request is *Exit All*
   8.3. **Exit Navigation Request Process** – Navigation Request is valid

9. **Case:** Navigation Request is *Suspend*
   9.1. Exit Request is *Suspend*
   9.2. Sequencing Request is *Exit*
   9.3. **Exit Navigation Request Process** – Navigation Request is valid

10. **Case:** Navigation Request is *Suspend All*
    10.1. Exit Request is *Suspend All*
    10.2. Sequencing Request is *Exit All*
    10.3. **Exit Navigation Request Process** – Navigation Request is valid

11. **Case:** Navigation Request is *Abandon*
11.1. Exit Request is Abandon
11.2. Sequencing Request is Exit
11.3. Exit Navigation Request Process – Navigation Request is valid

12. Case: Navigation Request is Abandon All
12.1. Exit Request is Abandon All
12.2. Sequencing Request is Exit All
12.3. Exit Navigation Request Process – Navigation Request is valid

13. Exit Navigation Request Process – Navigation Request is not valid

**NB.3 Notes/Changes from IMS SS Public Draft**

This model is identical in functionality and intent with that described in the IMS SS Public Draft. Descriptions and terminology have been simplified. Inconsistencies and partially specified behaviors have been fully specified.

References to SS data, application profile–specific behaviors and other descriptions not pertaining to just the navigation behavior have been eliminated. This part just describes the navigation behavior itself, not how the navigation process is used within the overall sequencing framework.

The following behaviors have been changed.

- The Exit (Exit, Suspend, Abandon) navigation behaviors specify both an Exit request and a Navigation Request. Sequencing requests assume that the appropriate activities have already been terminated. The proper termination activity is associated with a navigation request; thus, the request description is modified.

**NB.4 Open Issues**

The navigation process does not describe how to deal with activity threads. It makes sense to process the navigation request for the entire activity path for the main thread. How to process other threads is not obvious. [SS.NB.01]

Should the thread be specified, or taken from the thread assigned to the activity specified in the navigation request? [SS.NB.02]

The distinction between a navigation request and a navigation event may not be clear. SS deals with navigation requests. While there may be a one-to-one mapping between the requests and events, events imply a user interface or interaction model that is out of scope. [SS.NB.03]

Should there be a Resume navigation request to differentiate between a new attempt (one that abandons a previously suspended event) and just resuming? There appears to be no description of how or when a basic Continue, Previous or Choice request is for a new attempt versus resumption of a suspended attempt. [SS.NB.04]

Previous and Choice requests seem to imply that the current activity is always completed, never suspended. Suspend is always with an exit, never a directional navigation. Are there logical flow combinations that are useful but that cannot be expressed unless Suspend and Exit are made attributes of requests instead of fully independent requests? [SS.NB.05]

The difference between how to exit an activity as part of navigation and how to exit the activity as part of sequencing remains unclear. [SS.NB.06]

The extended descriptions of the exit behaviors in the table of navigation requests is duplicative with the definitions in the exit process model and could be deleted. [SS.NB.07]

There may be inconsistencies between using Choice versus Choose in describing both sequencing and navigation requests. [SS.NB.08]
The terminology of “exit” versus “terminate” for an activity may be confused throughout all parts of the documents. [SS.NB.09]

The Suspend All and Abandon All navigation requests are defined to process an activity and all of its parents. Is there the need for a request that processes all activities in the entire tree, or all activities for a particular sequencing thread. [SS.NB.10]

**NB.5 Implementation Recommendations**

Values of Completion and Measure are normalized between 0..1. Testing for the exact value of 1.0 is unreliable. Any value > 0.99999 should be considered equal to 1.0 in evaluating rule conditions (based on the required precision of 5 decimal digits).

Normalized data values are described as value space real. A decimal data type should be considered as an alternative to a float data type for the implementation to maintain the required precision.

Rule processors should use short circuit evaluation techniques when determining if the rule conditions are true. For example, assume an And rule contains a condition on both Objective Data Status and Objective Satisfied Status. If the Objective Data Status evaluates to False, the condition on the the value of Objective Satisfied Status should not be evaluated (in this case, the Objective Satisfied Status value is unreliable). The rule processor may reorder conditions to enable short circuit evaluation. However, short circuit evaluation is not required, as the default values insure that the same navigation behaviors will result whether short circuit evaluation is not used or not.

The implementation must ensure that data values are consistent when being accessed, i.e., the values of the set of activities that participate in a child set must be consistent while the entire set of rules applied during navigation are evaluated. Implementing proper transactional controls are left to the implementation.

**NB.6 Bindings**

Not applicable to a behavior model.

**NB.7 Navigation Process Service**

The navigation process may be implemented as a service. The navigation process has one service access point (SAP) for the Navigation Request Process.

**NB.7.1 Navigation Request Process SAP**

**Request:**
The Navigation Request Process requires data items to specify:
- The designation of the activity tree.
- The root activity of the tree for navigation. An Xpath expression specifying the root node as a part of a larger tree. Specifying a root permits navigation to be applied to part of a larger tree structure. The root for navigation may be the same as the root of the activity tree.
- The navigation request (from the Navigation Request vocabulary).
- A navigation location. An Xpath expression specifying a node in the activity tree. Optional (only for Choice requests).
- The sequencing thread. An integer.
- The learner.

**Process:**
The Navigation Request Process is defined above (section NB.2.1).
Results:
The Navigation Request Process returns the following upon completion:

- A Boolean indication if the Navigation Request Process was successful.
- An indication of the corresponding exit request if the process was successful (from the Exit Request vocabulary).
- An indication of the corresponding sequencing request if the process was successful (from the Sequencing Request vocabulary).
- An error indication if the process was not successful. Possible errors include:
  - Data errors:
    - The activity tree was not properly specified.
    - The root of the tree was not properly specified.
    - The Navigation Request was not properly specified.
    - The navigation location was not properly specified.
    - The thread was not properly specified (e.g., not 1, or there was no path from the root to the end activity for the specified thread).
    - The learner was not properly specified.
- Processing Errors:

The process description does not currently define how to return multiple error indicators.

Related Services:
The Navigation Request Process relies on the following other services:

- None.
EB. Exit Behavior Model (Just Exit V0.5)

The SS process delivers activities and tracks the current activity and other activities that have been attempted but not completed. Activities finish or are terminated due to other events. When the activity finishes, the triggering event may imply that one or more other activities must also terminate or exit. The sequencing rules may imply that a pending sequencing request is replaced by the sequencing request specified in the sequencing rule. The process of evaluating exit sequencing rules, recording information about the state of the activity when it finishes and terminating other activities based on the defined sequencing behaviors is called the “exit process”.

The exit process makes no assumptions as to how or when an exit request is generated.

The overall sequencing process (see Overall Sequencing Process Document OP) relates the exit process to the navigation, sequencing, delivery, and rollup processes.

The exit process is controlled by parts of the sequencing definition model:

- **Sequencing Rule Definitions** – rules applied to an activity that are used to specify sequencing behaviors for the activity.
- **Delivery Controls** – actions applied to set objective and progress data when an activity terminates.

The exit process uses parts of the tracking model:

- **Objective Information** – information about the results of the learner’s interactions related to an objective.
- **Progress Information** – information about a learner’s attempt at an activity, updated by the exit process.

The exit process uses data from the activity state model:

- **Activity State Information** – information about the results of the learner’s state for an activity.

The behavior of the exit process is defined in terms of a single process and two associated subprocesses:

- **Exit Request Process** – processes the actual exit request to set the corresponding state and activity data.
- **Completion Subprocess** – processes completion data when the sequencing engine, not content objects, controls completion.
- **Sequence Exit Rules Subprocess** – evaluates sequencing rules on activity exit and identifies the resulting exit and sequencing requests.

### EB.1 Exit Requests

The exit process responds to one of a set of different exit requests. An exit request defines a set of actions to be performed when the request is processed. The request names are tokens in a vocabulary. The names have no semantics or meanings themselves. The definition of the action is the complete definition of the required behavior.

<table>
<thead>
<tr>
<th>Exit Request</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continue</td>
<td>The current attempt on the currently delivered activity is terminated normally; the attempt is over.</td>
</tr>
<tr>
<td>Continue from Parent</td>
<td>The current attempt on the currently delivered activity and its parent are terminated normally; the attempts are over.</td>
</tr>
<tr>
<td>Previous</td>
<td>The current attempt on the currently delivered activity is terminated normally; the attempt is over.</td>
</tr>
<tr>
<td>Exit</td>
<td>The current attempt on the currently delivered activity is terminated normally; the attempt is over.</td>
</tr>
<tr>
<td>Exit Parent</td>
<td>The current attempt on the currently delivered activity and its parent are terminated normally; the attempts are over.</td>
</tr>
<tr>
<td>Exit Target</td>
<td>The current attempts on a subset of the current activities (from the target activity to...</td>
</tr>
</tbody>
</table>
Exit All
The current attempts on the current activities (from the root to the currently delivered activity) are terminated normally; the attempts are over.

Retry
The current attempt on the currently delivered activity is terminated normally; the attempt is over.

Retry All
The current attempt on the current activities (from the root to the currently delivered activity) are terminated normally; the attempts are over.

Suspend
The current attempt on the currently delivered activity is terminated normally; the attempt is not over, and the activity is not complete. The activity may be resumed at some time in the future (resumption is not a new attempt). The activity is marked as suspended in the activity state model.

Suspend All
The current attempt on the currently delivered activity and all of its parents are terminated normally; the attempts are not over, and the activities are not complete. The activities may be resumed at some time in the future (resumption is not a new attempt). The activities are marked as suspended in the activity state model.

Abandon
The current attempt on the currently delivered activity is terminated abnormally and the activity is not complete. The activity attempt may not be resumed. There is no rollback of any tracking data.

Abandon All
The current attempt on the currently delivered activity and all of its parents are terminated abnormally and the activities are not complete. The activity attempts may not be resumed. There is no rollback of any tracking data.

EB.2 Exit Behavior

The exit behavior describes how an exit processor interprets an exit request to identify an alternative sequencing request, to mark activities as terminated and to set activity progress and completion data.

An implementation must be capable of representing the processes described and have the implemented process exhibit the behavior described. There are no additional requirements on implementing the exit behavior model.

The exit behavior relies on the data descriptions from the sequencing definition model (see Sequencing Definition Model Document SM), the tracking model (see Tracking Model Document TM) and the activity state model (see Activity State Model Document AM). These data models also specify default data values or data inheritance in the activity tree that govern the access to activity, tracking, or state data.

EB.2.1 Sequencing Exit Rules Subprocess

Sequencing rules may be applied when an activity exits. These sequencing rules may override the exit and sequencing requests generated from a navigation request. The Sequence Exit Rules Subprocess evaluates sequencing rules and identifies the resulting exit and sequencing requests.

Sequencing rule conditions are specified by the Sequencing Rule Definitions. Evaluating the sequencing rules uses data values for Objective and Activity Progress attributes of the tracking model.

1. The subprocess is called when the current activity exits and evaluates only the sequencing rules whose actions might result in a new sequencing or exit request. The rules are evaluated in order, and the process stops at the first rule that evaluates to true.

The Sequence Exit Rules Subprocess evaluates sequencing rules for the identified activity and may optionally return exit and sequencing requests determined by the rule action.
The Sequence Exit Rule Subprocess is specified by the following pseudo code. The pseudo code describes only the exit rule processing. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The Sequence Exit Rules Subprocess only describes the expected behavior that an implementation will exhibit.

**Sequencing Exit Rules Behavior** (for the identified activity).

1. **If** there are Sequencing Rules for the activity where the attribute value **Rule Action** is **Continue** or **Continue From Parent or Previous or Exit Parent or Exit All or Retry or Retry All or Ignore** **Then**
   1.1. **For** each Sequencing Rule for the activity where the attribute value **Rule Action** is as listed above
       1.1.1. Initialize **Activity Condition Bag** as an empty collection
       1.1.2. **For** each **Rule Condition** for the Sequencing Rule for the activity
           1.1.2.1. Evaluate the **Rule Condition** for the rule by applying the Sequencing Rule attribute value **Rule Condition** for the activity to the appropriate corresponding data (either a value from the objective associated with the activity, a progress value for the activity, or a Limit Condition attribute value for the activity)
           1.1.2.2. **If** the **Rule Condition Operator** for the rule is **Not** **Then**
               1.1.2.2.1. Negate the Rule Condition evaluation result
           1.1.2.3. **If** the Rule Condition for the rule evaluates to **True** **Then**
               1.1.2.3.1. Add a **True** value to the **Activity Condition Bag**
           1.1.2.4. **If** the Rule Condition for the rule evaluates to **False** **Then**
               1.1.2.4.1. Add a **False** value to the **Activity Condition Bag**
       1.1.3. **If** the **Activity Children Status Bag** is empty **Then**
           1.1.3.1. **Exit** Sequencing Exit Rules Subprocess
       1.1.4. **Apply** the combination (**And, Any**) from the Sequencing Rule attribute value **Rule Combination** for the activity to the **Activity Condition Bag** to produce a single combined result for the Rule Conditions of the rule of the activity
       1.1.5. **If** the combination condition evaluates to **True** **Then**
           1.1.5.1. **Sequencing Request** is the attribute value **Rule Action** of the rule
           1.1.5.2. **Exit Request** is the attribute value **Rule Action** of the rule
           1.1.5.3. **Exit** Sequencing Exit Rules Process

2. **Exit** Sequencing Exit Rules Process

**EB.2.2 Completion Subprocess**

The result of a learner’s interactions with an activity is reflected in a set of objective and activity progress data. SS makes no assumptions as to how or when these results are set. A content object may directly set the appropriate objective and activity progress data. However, not all content objects may have the ability to set the data. The delivery controls **Completion Set by Content** and **Objective Set by Content** are used to indicate whether the content object will or will not set the data values in the tracking model used to control the sequencing behavior.

The Completion Subprocess uses data values from the Delivery Controls. It sets data values for Objective and Activity Progress attributes of the tracking model.

1. The Completion Subprocess specifies values to use when content object does not set values. Values are only set when the activity is being Tracked.
   a. An activity that terminates is considered complete.
   b. The objective associated with the activity is considered satisfied (but without a satisfaction measure).
2. These values can then participate in rollup.

Setting of values in the tracking model is controlled only by the delivery controls. Values set by the sequencing process overwrite any values set by the content object. If the delivery controls indicate that the exit process should not set values, and if the content object also does not set values, the default values from the tracking model are used in subsequent rollup and sequencing.
When an activity terminates or is suspended, actions are required to properly accumulate the time spent on the activity and the time spent on the attempt at the activity. Determining and accumulating such times could be done as part of the described completion subprocess (time is accumulated independently only if the activity is Tracked). This behavior is not described here, but an implementation must record and update the attempt and activity duration and wall clock times. The meaning of these data attributes (and the implicit behavior required to accumulate the times) are described in the tracking model (see Tracking Model Document TM).

The process does not explicitly return a result.

The Completion Subprocess is specified by the following pseudo code. The pseudo code describes only the Completion Subprocess. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The Completion Subprocess only describes the expected behavior that an implementation will exhibit.

Completion Behavior (for the identified activity).
1. **If** the Delivery Control attribute value Tracked for the activity is True **Then**
   1.1. **If** the Delivery Control attribute value Completion Set by Content for the activity is False **Then**
      1.1.1. Set the Activity Progress attribute value Activity Completion Status for the activity to True
      1.1.2. Set the Activity Progress attribute value Activity Completion Amount for the activity to 1.0
      1.1.3. Set the Activity Progress attribute value Activity Attempt Completion Status for the activity to True
      1.1.4. Set the Activity Progress attribute value Activity Attempt Completion Amount for the activity to 1.0
   1.2. **If** the Delivery Control attribute value Objective Set by Content for the activity is False **Then**
      1.2.1. Set the Objective attribute value Objective Data Status for the objective associated with the activity to True
      1.2.2. Set the Objective attribute value Objective Satisfied Status for the objective associated with the activity to True

**EB.2.3 Exit Request Process**

The Exit Request Process processes the exit request, processes exit sequencing rules and updates the activity state information for multiple activities.

The process uses the Sequencing Exit Rules Subprocess and the Completion Subprocess. It updates values in the activity state model.

1. For certain exit requests, the activity is exited and the sequencing rules associated with activity exit are evaluated. The rules may override the original exit and sequencing requests. For each specified exit request, appropriate activity progress and suspend states are set for the activities and the Completion Subprocess is applied to the activities.

The Exit Request Process either validates the exit request and optionally returns a new sequencing request or returns an error.

The Exit Request Process for an exit request is specified by the following pseudo code. The pseudo code describes only the exit request processing. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The Exit Request Process only describes the expected behavior that an implementation will exhibit.

Exit Request Behavior.
1. **If** there is no current activity **Then**
   1.1. Exit Exit Request Process – Error
2. **If** the Exit Request is not either Suspend or Suspend All **Then**
   2.1. Apply the Sequencing Exit Rules Subprocess to the current activity
2.2. **If the Sequencing Exit Rules Subprocess fails Then**

2.2.1. **Exit Exit Request Process – Error**

3. **Case:** Exit Request is **Continue**

3.1. Set the Activity Status attribute value *Activity in Progress* for the current activity to *False*

3.2. Set the Activity Status attribute value *Activity Suspended* for the current activity to *False*

3.3. Apply the Completion Subprocess to the current activity

3.4. **Exit Exit Request Process – Exit Request is valid**

4. **Case:** Exit Request is **Continue from Parent**

4.1. Set the Activity Status attribute value *Activity in Progress* for the current activity to *False*

4.2. Set the Activity Status attribute value *Activity Suspended* for the current activity to *False*

4.3. Apply the Completion Subprocess to the current activity

4.4. Set the Activity Status attribute value *Activity in Progress* for the parent of the current activity to *False*

4.5. Set the Activity Status attribute value *Activity Suspended* for the parent of the current activity to *False*

4.6. Apply the Completion Subprocess to the parent of the current activity

4.7. **Exit Exit Request Process – Exit Request is valid**

5. **Case:** Exit Request is **Previous**

5.1. Set the Activity Status attribute value *Activity in Progress* for the current activity to *False*

5.2. Set the Activity Status attribute value *Activity Suspended* for the current activity to *False*

5.3. Apply the Completion Subprocess to the current activity

5.4. **Exit Exit Request Process – Exit Request is valid**

6. **Case:** Exit Request is **Exit**

6.1. Set the Activity Status attribute value *Activity in Progress* for the current activity to *False*

6.2. Set the Activity Status attribute value *Activity Suspended* for the current activity to *False*

6.3. Apply the Completion Subprocess to the current activity

6.4. **Exit Exit Request Process – Exit Request is valid**

7. **Case:** Exit Request is **Exit Parent**

7.1. Set the Activity Status attribute value *Activity in Progress* for the current activity to *False*

7.2. Set the Activity Status attribute value *Activity Suspended* for the current activity to *False*

7.3. Apply the Completion Subprocess to the current activity

7.4. Set the Activity Status attribute value *Activity in Progress* for the parent of the current activity to *False*

7.5. Set the Activity Status attribute value *Activity Suspended* for the parent of the current activity to *False*

7.6. Apply the Completion Subprocess to the parent of the current activity

7.7. **Exit Exit Request Process – Exit Request is valid**

8. **Case:** Exit Request is **Exit Target**

8.1. Form the *Activity Path* as an ordered sequence of all activities from the current activity to the target activity

8.2. **If the Activity Path is empty Then**

8.2.1. **Exit Exit Request Process – Exit request not valid**

8.3. **For each activity on the Activity Path**

8.3.1. Set the Activity Status attribute value *Activity in Progress* for the activity to *False*

8.3.2. Set the Activity Status attribute value *Activity Suspended* for the activity to *False*

8.3.3. Apply the Completion Subprocess to the activity

8.4. **Exit Exit Request Process – Exit Request is valid**

9. **Case:** Exit Request is **Exit All**

9.1. Form the *Activity Path* as an ordered sequence of all activities from the current activity to the root of the activity tree

9.2. **If the Activity Path is empty Then**

9.2.1. **Exit Exit Request Process – Exit request not valid**

9.3. **For each activity on the Activity Path**

9.3.1. Set the Activity Status attribute value *Activity in Progress* for the activity to *False*

9.3.2. Set the Activity Status attribute value *Activity Suspended* for the activity to *False*

9.3.3. Apply the Completion Subprocess to the activity

9.4. **Exit Exit Request Process – Exit Request is valid**

10. **Case:** Exit Request is **Retry**

10.1. Set the Activity Status attribute value *Activity in Progress* for the current activity to *False*

10.2. Set the Activity Status attribute value *Activity Suspended* for the current activity to *False*
10.3. Apply the *Completion Subprocess* to the current activity
10.4. **Exit** Exit Request Process – Exit Request is valid

11. **Case:** Exit Request is *Retry All*
   11.1. Form the *Activity Path* as an ordered sequence of all activities from the current activity to the root of the activity tree
   11.2. **If** the *Activity Path* is empty **Then**
       11.2.1. **Exit** Exit Request Process – Exit request not valid
   11.3. **For** each activity on the *Activity Path*
       11.3.1. Set the Activity Status attribute value *Activity in Progress* for the current activity to *False*
       11.3.2. Set the Activity Status attribute value *Activity Suspended* for the current activity to *False*
       11.3.3. Apply the *Completion Subprocess* to the activity
   11.4. **Exit** Exit Request Process – Exit Request is valid

12. **Case:** Exit Request is *Suspend*
   12.1. Set the Activity Status attribute value *Activity in Progress* for the current activity to *False*
   12.2. Set the Activity Status attribute value *Activity Suspended* for the current activity to *True*
   12.3. **Exit** Exit Request Process – Exit Request is valid

13. **Case:** Exit Request is *Suspend All*
   13.1. Form the *Activity Path* as an ordered sequence of all activities from the current activity to the root of the activity tree
   13.2. **If** the *Activity Path* is empty **Then**
       13.2.1. **Exit** Exit Request Process – Exit request not valid
   13.3. **For** each activity on the *Activity Path*
       13.3.1. Set the Activity Status attribute value *Activity in Progress* for the current activity to *False*
       13.3.2. Set the Activity Status attribute value *Activity Suspended* for the current activity to *True*
   13.4. **Exit** Exit Request Process – Exit Request is valid

14. **Exit** Exit Request Process – Exit Request is not valid

**EB.3 Notes/Changes from IMS SS Public Draft**

This is a new data behavior model. Exit processing is implicit in the IMS SS Public Draft with aspects intermixed within both the navigation and sequencing request processes. This model makes the behavior explicit so it can be referenced and used in a consistent manner throughout the rest of SS.

References to SS data, application profile–specific behaviors and other descriptions not pertaining to just the exit behavior are not included. This part just describes the exit behavior itself, not how the exit behavior process is used within the overall sequencing framework.

**EB.4 Open Issues**

The exit process does not describe how to deal with activity threads. It makes sense to process the exit request for the entire activity path for the main thread. How to process other threads is not obvious. [SS.EB.01]

Should the thread be specified, or taken from the thread assigned to the activity specified in the exit request? [SS.EB.02]

Are the exit requests all really different, or can they be collapsed into a smaller set of requests? [SS.EB.03]

Should completion behavior be applied to only the current activity or to all activities that are processed in the exit request? [SS.EB.04]

The public draft says you can only suspend an activity, not its parent. Suspend All requires that you suspend an activity and its parents. If you then exit, and issue a Choice request to some other location, you can create the equivalent of suspending and resuming from a parent. What is the intent? [SS.EB.05]
Exit should also deal with how to accumulate time. This is not included in the current descriptions. [SS.EB.06]

The behavior is based on the assumption that there is no particular need to explicitly evaluate the sequencing rules when any activity terminates. In particular, prior to rollup, there is no need to explicitly trigger post conditions independently of a sequencing request. The assumption is that the navigation behavior will produce a sequencing request that will be processed after rollup, and that sequencing request will evaluate all of the sequencing rules, including the post condition rules. Alternatively, if the assumption is not true, there could be an additional exit/sequencing process to process the post condition rules for either an individual activity or the set of activities on the identified activity path. [SS.EB.07]

The Exit All, Retry All, Suspend All and Abandon All exit requests are defined to process an activity and all of its parents. Is there the need for a request that processes all activities in the entire tree, not just on the path from the activity to the root, or to process all activities for a particular sequencing thread? [SS.EB.08]

The behavior of a “from parent” exit request is not specified if there is no parent of the activity. [SS.EB.09]

**EB.5 Implementation Recommendations**

Values of Completion and Measure are normalized between 0..1. Testing for the exact value of 1.0 is unreliable. Any value > 0.99999 should be considered equal to 1.0 in evaluating rule conditions (based on the required precision of 5 decimal digits).

Normalized data values are described as value space real. A decimal data type should be considered as an alternative to a float data type for the implementation to maintain the required precision.

Rule processors should use short circuit evaluation techniques when determining if the rule conditions are true. For example, assume an And rule contains a condition on both Objective Data Status and Objective Satisfied Status. If the Objective Data Status evaluates to False, the condition on the the value of Objective Satisfied Status should not be evaluated (in this case, the Objective Satisfied Status value is unreliable). The rule processor may reorder conditions to enable short circuit evaluation. However, short circuit evaluation is not required, as the default values insure that the same navigation behaviors will result whether short circuit evaluation is not used or not.

The implementation must ensure that data values are consistent when being accessed, i.e., the values of the set of activities that participate in a child set must be consistent while the entire set of rules applied during exit are evaluated. Implementing proper transactional controls are left to the implementation.

**EB.6 Bindings**

Not applicable to a behavior model.

**EB.7 Exit Process Service**

The exit process may be implemented as a service. The exit process has one service access point (SAP) for the Exit Request Process.

**EB.7.1 Exit Request Process SAP**

**Request:**

The Exit Request Process requires data items to specify:

- The designation of the activity tree.
- The root activity of the tree for sequencing. An Xpath expression specifying the root node as a part of a larger tree. Specifying a root permits exit to be applied to part of a larger tree structure. The root for exit may be the same as the root of the activity tree.
• The exit request (from the Exit Request vocabulary).
• A current activity. An XPath expression specifying a node in the activity tree.
• The target activity (optional, only for Exit Target requests).
• The sequencing thread. An integer.
• The learner.

Process:
The Exit Request Process is defined above (section EB.2.2).

Results:
The Exit Request Process returns the following upon completion:
• A Boolean indication if the Exit Request Process was successful.
• An indication of the overriding sequencing request, optional (from the Sequencing Request vocabulary).
• An error indication if the process was not successful. Possible errors include:
  o Data errors:
    ▪ The activity tree was not properly specified.
    ▪ The root of the tree was not properly specified.
    ▪ The current activity was not properly specified.
    ▪ The target activity was not properly specified.
    ▪ The Exit Request was not properly specified.
    ▪ The thread was not properly specified (e.g., not 1, or there was no path from the root to the end activity for the specified thread).
    ▪ The learner was not properly specified.
  o Processing Errors:
    ▪ Error in the Exit Request Process.

The process description does not currently define how to return multiple error indicators.

Related Services:
The Exit Request Process relies on the following other services:
• Tracking Model Services (used to access and set values in the Tracking Model).
• Activity State Model Services (used to access and set values in the Activity State Model).
SB. Sequencing Behavior Model (Just Sequencing V0.6)

The SS process is driven in part by a “sequencing request” – an externally generated request to identify an element of content for eventual delivery to the learner. The SS process evaluates the request in terms of the content model described by the activity tree, the learner’s current location in the content experience, and determines what content object should be delivered to the learner. This is essentially a mapping or resolution process that converts a sequencing request into the identification of a content object. The process of evaluating the request and identifying the content object (or returning an error) is called the “sequencing process”.

The sequencing process makes no assumptions as to how or when a sequencing request is generated. The sequencing process must insure that sequencing requests are valid. Additionally, the sequencing process makes no assumptions about how or when the identified content object is actually delivered to the learner. In most cases, a delivery request for the identified content object will be passed immediately to the delivery service. As part of identifying the content object, the sequencing process uses elements of the tracking model and overall activity tree structure.

The overall sequencing process (see Overall Sequencing Process Document OP) relates the sequencing process to the navigation, delivery, exit, and rollup processes.

The sequencing process is controlled by parts of the sequencing definition model:
- **Sequencing Control Modes** – controls for types of sequencing requests that may apply to a collection of activities.
- **Sequencing Rule Definitions** – rules, applied to an activity, that are used to specify sequencing behaviors for the activity.
- **Limit Conditions** – limits on how many times, how long and when an activity is available.
- **Sequencing Thread Definitions** – specifications of parallel activity threads.
- **Objective Description** – the learning objective associated with an activity.

The sequencing process uses all parts of the tracking model:
- **Objective Information** – information about the results of the learner’s interactions related to an objective.
- **Progress Information** – information about a learner’s attempt at an activity.

The sequencing process uses data from the activity state model:
- **Activity State Information** – information about the results of the learner’s state for an activity.

The behavior of the sequencing process is defined in terms of ten processes (one for each sequencing request), an overall sequencing process and six associated subprocesses:
- **Sequencing Rules Check Subprocess** – applies sequencing rules to determine if the action of a single sequencing rule action should be performed.
- **Limit Conditions Check Subprocess** – applies limit conditions to determine if an activity is a candidate for delivery.
- **Flow Tree Traversal Subprocess** – determines the next activity in depth-first traversal of the activity tree.
- **Flow Activity Sequencing Subprocess** – determines if a single activity should be delivered by checking limit conditions and sequencing rules, optionally traversing the activity tree to the next activity.
- **Check Activity Subprocess** – determines if a single activity should be delivered by checking limit conditions and sequencing rules, ignoring control modes.
- **Choice Activity Sequence Subprocess** – determines if a Choice sequencing request is permitted for a single activity by checking limit conditions and sequencing rules.
- **Start Sequencing Request Process** – processes a Start sequencing request.
- **Continue Sequencing Request Process** – processes a Continue sequencing request.
- **Continue From Parent Sequencing Request Process** – processes a Continue From Parent sequencing request.
• Previous Sequencing Request Process – processes a Previous sequencing request.
• Choice Sequencing Request Process – processes a Choice sequencing request.
• Retry Sequencing Request Process – processes a Retry sequencing request.
• Retry All Sequencing Request Process – processes a Retry All sequencing request.
• Exit Sequencing Request Process – processes an Exit sequencing request.
• Exit Parent Sequencing Request Process – processes an Exit Parent sequencing request.
• Exit All Sequencing Request Process – processes an Exit All sequencing request.

These individual processes are part of the overall Sequencing Request Process that controls all sequencing behavior.

The sequencing process traverses the activity tree, in depth-first order (or reverse). Tree traversal continues until the traversal identifies a termination point or an activity to deliver. Identifying the activity to deliver is always constrained by limit conditions.

SB.1 Sequencing Requests

The sequencing process responds to one of a set of different sequencing requests. A sequencing request defines a set of actions to be performed when the request is processed. The request names are tokens in a vocabulary. The names have no semantics or meanings themselves. The definition of the action is the complete definition of the required behavior.

<table>
<thead>
<tr>
<th>Sequencing Request</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start</strong></td>
<td>Start the sequencing process at the root of the activity tree. Traverse the tree in depth-first order and evaluate the activities:</td>
</tr>
<tr>
<td></td>
<td>• If the activity has only a “Choice” sequencing control mode (value “True”), exit without identifying an activity to deliver – the system waits for another navigation request.</td>
</tr>
<tr>
<td></td>
<td>• If the activity does not have a “Flow” sequencing control mode (value “False”), exit without identifying an activity to deliver – the sequencing system waits for another navigation request.</td>
</tr>
<tr>
<td></td>
<td>• If the tree traversal reaches a leaf activity, the activity is the candidate to deliver to the learner.</td>
</tr>
<tr>
<td><strong>Continue</strong></td>
<td>A Continue sequencing request is processed only if the “Flow” sequencing control mode (value “True”) is enabled for the parent of the current activity; otherwise, this sequencing request has no effect.</td>
</tr>
<tr>
<td></td>
<td>Start the sequencing process at the current activity. Traverse the tree in depth-first order and evaluate the activities:</td>
</tr>
<tr>
<td></td>
<td>• If the activity has only a “Choice” sequencing control mode (value “True”), exit without identifying an activity to deliver – the sequencing system waits for another navigation request.</td>
</tr>
<tr>
<td></td>
<td>• If the activity does not have a “Flow” sequencing control mode (value “False”), exit without identifying an activity to deliver – the sequencing system waits for another navigation request.</td>
</tr>
<tr>
<td></td>
<td>• If the tree traversal reaches a leaf activity, the activity is the candidate to deliver to the learner.</td>
</tr>
<tr>
<td><strong>Continue From Parent</strong></td>
<td>A Continue From Parent sequencing request is processed only if the “Flow” sequencing control mode (value “True”) is enabled for the parent of the current activity (the cluster); otherwise, this sequencing request has no effect.</td>
</tr>
<tr>
<td></td>
<td>Start the sequencing process at the parent of the current activity. Traverse the tree in depth-first order and evaluate the activities:</td>
</tr>
<tr>
<td></td>
<td>• If the activity has only a “Choice” sequencing control mode (value “True”), exit without identifying an activity to deliver – the sequencing system waits for another navigation request.</td>
</tr>
</tbody>
</table>
for another navigation request.

• If the activity does not have a “Flow” sequencing control mode (value “False”), exit without identifying an activity to deliver – the sequencing system waits for another navigation request.
• If the tree traversal reaches a leaf activity, the activity is the candidate to deliver to the learner.

**Previous**

A Previous sequencing request is processed only if a “Flow” sequencing control mode is enabled (value “True”) for the parent of the current activity (the cluster); otherwise, this sequencing request has no effect.

A Previous sequencing request is processed only if the “Forward Only” sequencing control mode for the current activity is “False”.

Start the sequencing process at the current activity. Traverse the tree in *reverse* depth-first order and evaluate the activities:

• If the activity has only a “Choice” sequencing control mode (value “True”), exit without identifying an activity to deliver – the sequencing system waits for another navigation request.
• If the activity does not have a “Flow” sequencing control mode (value “False”), exit without identifying an activity to deliver – the sequencing system waits for another navigation request.
• If the tree traversal reaches a leaf activity, the activity is the candidate to deliver to the learner.

**Choice**

This sequencing request is accompanied by a “target” of the request, the activity that is requested as the next activity in the sequence.

If the parent of the current activity (the cluster) does not have a “Choice” sequencing control mode (value “True”), exit without identifying an activity to deliver – the sequencing system waits for another navigation request.

If the parent of the target activity (the cluster) does not have a “Choice” sequencing control mode (value “True”), exit without identifying an activity to deliver – the sequencing system waits for another navigation request.

Find the first common ancestor activity of the current activity (or the root if there is no current activity) and the target activity. Form a path from the current activity to the common ancestor activity and then to the target activity. Start the sequencing process at the current activity. Traverse the path from the current activity to the target activity and evaluate the activities (each step in the traversal may be *forward* or *reverse* relative to the depth-first tree traversal):

• If traversal is in the *forward* direction, and a *Stop Forward Traversal* rule for the activity evaluates to “True”, exit without identifying an activity to deliver – the sequencing system waits for another navigation request.
• If traversal is in the *reverse* direction, and a *Forward Only* rule for the activity evaluates to “True”, exit without identifying an activity to deliver – the sequencing system waits for another navigation request.

**Retry**

The current activity is identified as the activity to deliver (recorded as a new attempt). The current activity should already be terminated.

**Retry All**

The current activity is identified as the activity to deliver (recorded as a new attempt on the entire path). The current activity and all its parents should already be terminated.

**Exit**

Advance to the next activity on activity exit if “Auto Advance” control is enabled. Start the sequencing process at the current activity.

• If the current activity is the root of the activity tree, exit sequencing.
• If the parent of the current activity (the cluster) has either a “Flow” or
“Choice” sequencing control mode, exit without identifying an activity to deliver – the sequencing system waits for another navigation request.

- If the parent of the current activity (the cluster) has an “Auto Advance” sequencing control mode (value “True”), traverse the tree in depth-first order and evaluate the activities.
  - If the activity has only a “Choice” sequencing control mode (value “True”), exit without identifying an activity to deliver – the sequencing system waits for another navigation request.
  - If the activity does not have a “Flow” sequencing control mode (value “False”), exit without identifying an activity to deliver – the sequencing system waits for another navigation request.
  - If the tree traversal reaches a leaf activity, the activity is the candidate to deliver to the learner.

<table>
<thead>
<tr>
<th>Exit Parent</th>
<th>If the current activity is the root of the activity tree, exit sequencing. Otherwise, exit without identifying an activity to deliver – the sequencing system waits for another navigation request. The current activity and its parent should already be terminated.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit All</td>
<td>Exit sequencing. The current activity and all its parents should already be terminated.</td>
</tr>
</tbody>
</table>

SB.2 Sequencing Behavior

The sequencing behavior describes how a sequencing processor interprets a sequencing request in combination with the elements of the sequencing definition model and with instance data from the tracking model and activity state model to validate the sequencing request and identify the content object for delivery.

An implementation must be capable of representing the processes described and have the implemented process exhibit the behavior described. There are no additional requirements on implementing the sequencing behavior model.

The sequencing behavior relies on the data descriptions from the sequencing definition model (see Sequencing Definition Model Document SM), the tracking model (see Tracking Model Document TM) and the activity state model (see Activity State Model Document AM). These data models also specify default data values or data inheritance in the activity tree that govern the access to activity or tracking data.

SB.2.1 Sequencing Rules Check Subprocess

Sequencing is controlled by the rules associated with each activity. Rules are evaluated throughout the tree traversal used in processing a sequencing request. The Sequencing Rules Check Subprocess is used to determine if a single rule action should be performed.

Sequencing rule conditions are specified by the Sequencing Rule Definitions. Evaluating the sequencing rule uses data values for Objective and Activity Progress attributes of the tracking model and Limit Condition attributes.

The Sequencing Rules Check Subprocess is applied to the single identified rule for the identified activity. It returns a Boolean: True if the rule condition evaluates to True and the corresponding rule action should be performed; False if the rule condition evaluates to False and the corresponding rule action should not be performed.

The Sequencing Rules Check Subprocess is specified by the following pseudo code. The pseudo code describes only the rule checking logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The Sequencing Rules Check Subprocess only describes the expected behavior that an implementation will exhibit.

**Sequencing Rules Check Behavior** (for the identified activity and sequencing rule).
1. Initialize Activity Condition Bag as an empty collection
2. For each Rule Condition for the Sequencing Rule for the activity
2.1. Evaluate the Rule Condition for the rule by applying the Sequencing Rule attribute value Rule Condition for the activity to the appropriate corresponding data (either a value from the objective associated with the activity, a progress value for the activity, or a Limit Condition attribute value for the activity)

2.2. If the Rule Condition Operator for the rule is Not Then
   2.2.1. Negate the Rule Condition

2.3. If the Rule Condition for the rule evaluates to True Then
   2.3.1. Add a True value to the Activity Condition Bag

2.4. If the Rule Condition for the rule evaluates to False Then
   2.4.1. Add a False value to the Activity Condition Bag

3. If the Activity Children Status Bag is empty Then
   3.1. Return False from Sequencing Rules Check Subprocess – no conditions

4. Apply the combination from the Sequencing Rule attribute value Rule Combination for the activity to the Activity Condition Bag to produce a single combined result for the Rule Conditions of the rule for the activity

5. If the combination condition evaluates to True Then
   5.1. Return True from Sequencing Rules Check Subprocess – rule applies

6. If the combination condition evaluates to False Then
   6.1. Return False from Sequencing Rules Check Subprocess – rule does not apply

SB.2.2 Limit Conditions Check Subprocess

Sequencing is controlled by limit conditions associated with each activity. Limit conditions are evaluated throughout the tree traversal used in processing a sequencing request. The Limit Conditions Check Subprocess is used to determine if an activity is a candidate for delivery. Activities whose limit conditions are violated are not candidates for delivery.

Limit conditions are specified by all of the Limit Condition attributes. These attributes constrain the number of access attempts, the duration of access or time limit on access, and the time frame of access. Limit conditions are evaluated using data values for Activity Progress attributes of the tracking model.

The Limit Conditions Check Subprocess is applied to the single identified activity. It returns a Boolean: True if any of the limit conditions are violated; or False if none of the limit conditions are violated.

The Limit Conditions Check Subprocess for an activity is specified by the following pseudo code. The pseudo code describes only the limit checking logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. Limit Conditions Check Subprocess only describes the expected behavior that an implementation will exhibit.

Limit Conditions Check Behavior (for the identified activity).
1. If the Limit Condition attribute value Limit Condition Attempt Control for the activity is True Then
   1.1. If the Activity Progress attribute value Activity Attempt Count for the activity is greater than or equal to the Limit Condition attribute value Limit Condition Attempt Limit for the activity Then
       1.1.1. Return True from Limit Conditions Check Subprocess – Activity Limit Exceeded

2. If the Limit Condition attribute value Limit Condition Activity WallClock Duration Control for the activity is True Then
   2.1. If the Activity Progress attribute value Activity WallClock Duration for the activity is greater than or equal to the Limit Condition attribute value Limit Condition Activity WallClock Duration Limit for the activity Then
       2.1.1. Return True from Limit Conditions Check Subprocess – Activity WallClock Duration Limit Exceeded

3. If the Limit Condition attribute value Limit Condition Activity Interaction Duration Control for the activity is True Then
   3.1. If the Activity Progress attribute value Activity Interaction Duration for the activity is greater than or equal to the Limit Condition attribute value Limit Condition Activity Interaction Duration Limit for the activity Then
3.1.1. **Return** True from *Limit Conditions Check Subprocess* – Activity Interaction Duration Limit Exceeded

4. **If** the Limit Condition attribute value *Limit Condition Activity Attempt WallClock Duration Control* for the activity is **True**
   4.1. **If** the Activity Progress attribute value *Activity Attempt WallClock Duration* for the activity attempt is greater than or equal to the Limit Condition attribute value *Limit Condition Activity Attempt WallClock Duration Limit* for the activity **Then**
   4.1.1. **Return** True from *Limit Conditions Check Subprocess* – Activity Attempt WallClock Duration Limit Exceeded

5. **If** the Limit Condition attribute value *Limit Condition Activity Attempt Interaction Duration Control* for the activity is **True**
   5.1. **If** the Activity Progress attribute value *Activity Attempt Interaction Duration* for the activity attempt is greater than or equal to the Limit Condition attribute value *Limit Condition Activity Attempt Interaction Duration Limit* for the activity **Then**
   5.1.1. **Return** True from *Limit Conditions Check Subprocess* – Activity Attempt Interaction Duration Limit Exceeded

6. **If** the Limit Condition attribute value *Limit Condition Time Limit Control* for the activity is **True**
   6.1. **If** the current timepoint is before the Limit Condition attribute value *Limit Conditions Begin Time Limit* for the activity **Then**
   6.1.1. **Return** True from *Limit Conditions Check Subprocess* – Activity End Time Limit not met

7. **Return** False from *Limit Conditions Check Subprocess* – No limits violated

### SB.2.3 Flow Tree Traversal Subprocess

Sequencing traverses the activity tree in depth-first order. Different sequencing requests traverse either forward or backward through the activity tree. The *Flow Tree Traversal Subprocess* is used to determine the next activity in traversal order, starting at a particular activity (or it determines the previous activity if traversing in the backward direction).

Traversal is controlled by sequencing rules and Control Modes. Sequencing rule conditions are specified by the Sequencing Rule Definitions. The process uses the *Sequencing Rules Check Subprocess*.

The traversal process is defined as a rule- and limit-checking traversal of the activity tree.

1. The *Flow Tree Traversal Subprocess* traverses the tree, either forward or backward, from the specified activity to identify the next activity (one single step in the tree traversal). Traversal beyond the end of the tree (in forward traversal) or the start of the tree (in backward traversal) is an exception. Only sequencing rules related to traversing the tree in a *Flow* control mode are applied.

The *Flow Tree Traversal Subprocess* is applied to the identified activity in the specified direction. If successful, it returns the *Next* activity in the tree in traversal order.

The *Flow Tree Traversal Subprocess* is specified by the following pseudo code. The pseudo code describes only the tree traversal logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The *Flow Tree Traversal Subprocess* only describes the expected behavior that an implementation will exhibit.

**Flow Tree Traversal Behavior** (for the identified activity and the specified direction).

1. **If** the activity is not the root activity of the activity tree **Then**
   1.1. **If** the Sequencing Control Mode attribute value *Flow* for the parent of the activity is **False** **Then**
      1.1.1. **Exit** *Flow Tree Traversal Subprocess* – No next activity found
   2. **If** the traversal direction is *forward* **Then**
      2.1. **If** the activity is the last activity in a depth-first, forward tree traversal **Then**
         2.1.1. **Exit** *Flow Tree Traversal Subprocess* – No next activity found
2.2. If there are Sequencing Rules with Rule Action attribute value of *Stop Forward Traversal* for the activity Then
   2.2.1. For each such Rule
      2.2.1.1. Apply the *Sequencing Rules Check Subprocess* to the rule and the activity
      2.2.1.2. If the *Sequencing Rules Check Subprocess* returns True Then
         2.2.1.2.1. Exit *Flow Tree Traversal Subprocess* – No next activity found
   2.3. Traverse the tree depth-first, forward one activity to the Next activity
   2.4. Exit *Flow Tree Traversal Subprocess* – Return the Next activity

3. If the traversal direction is *backward* Then
   3.1. If the activity is the root activity of the tree Then
      3.1.1. Exit *Flow Tree Traversal Subprocess* – No next activity found
   3.2. If the Sequencing Control Mode attribute value *Forward Only* for the parent of the activity is True Then
      3.2.1. Exit *Flow Tree Traversal Subprocess* – No next activity found
   3.3. Traverse the tree depth-first, backward one activity to the Previous activity
   3.4. Exit *Flow Tree Traversal Subprocess* – Return the Previous activity

**SB.2.4 Flow Activity Sequencing Subprocess**

Sequencing traverses the activity tree, applying sequencing rules and limit conditions. The *Flow Activity Sequencing Subprocess* is used to determine if an activity should be delivered, or if the traversal should examine the next activity in the tree. The process is applied to a single activity, checking both limit conditions and sequencing rules.

Traversal is controlled by sequencing rules and Control Modes. Sequencing rule conditions are specified by the Sequencing Rule Definitions. The process uses the *Sequencing Rules Check Subprocess*, the *Limits Conditions Check Subprocess*, and the *Flow Tree Traversal Subprocess*.

The process is defined as a rule- and limit-checking traversal of the activity tree.

1. The process traverses the entire tree, depth first, for the particular activity. If only *Choice* is enabled for the activity, stop without indentifying an activity to deliver. Check the limit conditions for the activity and stop if any limits are exceeded. Stop if the activity is disabled. Skip activities marked to be skipped, and traverse to the next activity in the specified direction. If the activity is not a leaf, traverse the tree to find the next activity.

The *Flow Activity Sequencing Subprocess* is applied to the identified activity in the specified direction. If successful, it returns the Next activity in the tree in traversal order.

The *Flow Activity Sequencing Subprocess* is specified by the following pseudo code. The pseudo code describes only the activity sequencing logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The *Flow Activity Sequencing Subprocess* only describes the expected behavior that an implementation will exhibit.

**Flow Activity Sequencing Behavior** (for the identified activity and the specified direction).
1. Apply the *Limit Condition Check Subprocess* to the activity
2. If the *Limit Condition Check Subprocess* returns True Then
   2.1. Exit *Flow Activity Sequencing Subprocess* – Nothing identified for delivery
3. If the Sequencing Control Mode attribute value *Choice* for the activity is True and the Sequencing Control Mode attribute value *Flow* for the candidate activity is False Then
   3.1. Exit *Flow Activity Sequencing Subprocess* – Nothing identified for delivery
4. If there are Sequencing Rules with Rule Action attribute value of *OK* for the activity Then
   4.1. Ignore the rules – OK rules do not influence sequencing
5. If there are Sequencing Rules with Rule Action attribute value of *Hidden from Choice* for the activity Then
   5.1. Ignore the rules – Hidden from Choice rules do not influence sequencing
6. If there are Sequencing Rules with Rule Action attribute value of *Disabled* for the activity Then
   6.1. For each such Rule
6.1.1. Apply the Sequencing Rules Check Subprocess for the rule and the activity

6.1.2. If the Sequencing Rules Check Subprocess returns True Then

6.1.2.1. Exit Flow Activity Sequencing Subprocess – Nothing identified for delivery

7. If there are Sequencing Rules with Rule Action attribute value of Skip for the activity Then

7.1. For each such Rule

7.1.1. Apply the Sequencing Rules Check Subprocess for the rule and the activity

7.1.2. If the Sequencing Rules Check Subprocess returns True Then

7.1.2.1. Apply the Flow Tree Traversal Subprocess in the specified direction to the activity

7.1.2.2. If the Flow Tree Traversal Subprocess did not identify the next activity in the tree Then

7.1.2.2.1. Exit Flow Activity Sequencing Subprocess – Nothing identified for delivery

7.1.2.3. Else

7.1.2.3.1. Exit Flow Activity Sequencing Subprocess – Next activity specified by Flow Tree Traversal Subprocess

8. If the activity is not a leaf node in the activity tree Then

8.1. Apply the Flow Tree Traversal Subprocess in the specified direction to the activity

8.1.1. If the Flow Tree Traversal Subprocess did not identify the next activity in the tree Then

8.1.1.1. Exit Flow Activity Sequencing Subprocess – Nothing identified for delivery

8.1.2. Else

8.1.2.1. Exit Flow Activity Sequencing Subprocess – Next activity specified by Flow Tree Traversal Subprocess

9. Exit Flow Activity Sequencing Subprocess – Activity is a leaf and candidate for delivery

SB.2.5 Check Activity Subprocess

Sequencing traverses the activity tree, applying sequencing rules and limit conditions. The Check Activity Subprocess is used to determine if an activity should be delivered. The process is applied to a single activity, checking both limit conditions and sequencing rules.

Traversal is controlled by sequencing rules. Sequencing rule conditions are specified by the Sequencing Rule Definitions. The process uses the Sequencing Rules Check Subprocess and the Limit Conditions Check Subprocess.

The process is defined as limit checking process.

1. Check the limit conditions for the activity and stop if any limits are exceeded. Stop if the activity is disabled.

The Check Activity Subprocess is applied to the single identified activity. It returns a Boolean: True if any of the limit conditions or rules are violated; or False if none of the limit conditions or rules are violated.

The Check Activity Subprocess is specified by the following pseudo code. The pseudo code describes only the checking logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The Check Activity Subprocess only describes the expected behavior that an implementation will exhibit.

Check Activity Behavior (for the identified activity).

1. Apply the Limit Condition Check Subprocess to the activity

2. If the Limit Condition Check Subprocess returns True Then

2.1. Return True from Check Activity Subprocess – Limits or rules violated

3. If there are Sequencing Rules with Rule Action attribute value of Disabled for the activity Then

3.1. For each such Rule

3.1.1. Apply the Sequencing Rules Check Subprocess for the rule and the activity

3.1.2. If the Sequencing Rules Check Subprocess returns True Then

3.1.2.1. Return True from Check Activity Subprocess – Limits or rules violated

4. Return False from Check Activity Subprocess – No limits or rules violated

SB.2.6 Choice Activity Sequence Subprocess
Sequencing traverses the activity tree, applying sequencing rules and limit conditions. The *Choice Activity Sequence Subprocess* is used to determine if a *Choice* sequencing request on an activity is permitted. The process is applied to a single activity, checking both limit conditions and sequencing rules.

Traversal is controlled by sequencing rules and Control Modes. Sequencing rule conditions are specified by the Sequencing Rule Definitions. The process uses the *Sequencing Rules Check Subprocess* and the *Limit Conditions Check Subprocess*.

The process is defined as a rule- and limit-checking process.

1. Check the limit conditions for the activity and stop if any limits are exceeded. Stop if the activity is disabled. Stop if traversal is not permitted (if there are Stop Forward Traversal rules for traversal in the forward direction or a Forward Only control mode for traversal in the backward direction).

The *Choice Activity Sequence Subprocess* is applied to the single identified activity. It returns a Boolean: True if any of the limit conditions or rules are violated; or False if none of the limit conditions or rules are violated.

The *Choice Activity Sequence Subprocess* is specified by the following pseudo code. The pseudo code describes only the checking logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The *Choice Activity Sequence Subprocess* only describes the expected behavior that an implementation will exhibit.

**Choice Activity Sequence Behavior** (for the identified activity).

1. Apply the *Limit Condition Check Subprocess* to the activity
2. **If** the *Limit Condition Check Subprocess* returns True **Then**
   2.1. **Return** True from *Choice Activity Sequence Subprocess* – Limits or rules violated
3. **If** there are Sequencing Rules with Rule Action attribute value of Disabled for the activity **Then**
   3.1. **For** each such Rule
       3.1.1. Apply the *Sequencing Rules Check Subprocess* for the rule and the activity
       3.1.2. **If** the *Sequencing Rules Check Subprocess* returns True **Then**
       3.1.2.1. **Return** True from *Choice Activity Sequence Subprocess* – Limits or rules violated
4. **If** the traversal direction is forward **Then**
   4.1. **If** there are Sequencing Rules with Rule Action attribute value of Stop Forward Traversal for the activity **Then**
       4.1.1. **For** each such Rule
           4.1.1.1. Apply the *Sequencing Rules Check Subprocess* to the rule and the activity
           4.1.1.2. **If** the *Sequencing Rules Check Subprocess* returns True **Then**
           4.1.1.2.1. **Return** True from *Choice Activity Sequence Subprocess* – Limits or rules violated
5. **If** the traversal direction is backward **Then**
   5.1. **If** there is a parent of the activity **Then**
       5.1.1. **If** the Sequencing Control Mode attribute value Forward Only for the parent of the activity is True **Then**
       5.1.1.1. **Return** True from *Choice Activity Sequence Subprocess* – Limits or rules violated
6. **Return** False from *Choice Activity Sequence Subprocess* – No limits or rules violated

**SB.2.7 Start Sequencing Request Process**

A Start sequencing request is used to identify the first activity to deliver. The *Start Sequencing Request Process* traverses the activity tree, applying sequencing rules and limit conditions to identify the first activity to deliver.

The sequencing process uses the *Flow Activity Sequencing Subprocess*. It does not explicitly use any sequencing definitions or tracking model data.
The sequencing request process is defined as a rule- and limit-checking traversal of the activity tree.

1. The process traverses the entire tree, forward, depth first, from the root using the Flow Activity Sequencing Subprocess. At any activity, if only Choice is enabled, stop without indentifying an activity to deliver. Check the limit conditions for the activity and stop if any limits are exceeded. Stop if the traversal reaches an activity that is disabled. Skip (and traverse forward) activities marked to be skipped. If the activity is not a leaf, traverse the tree to find the next activity. The first leaf activity found is the candidate activity to deliver.

The process can be described as a tree traversal loop. Start at the root, use the Flow Activity Sequencing Subprocess to determine if the loop should terminate, because either the activity should or should not be delivered. Otherwise, continue the loop with the next activity in the tree traversal.

The Start Sequencing Request Process either validates the sequencing request and returns the candidate activity to deliver or returns an indication that there is no activity to deliver.

The Start Sequencing Request Process is specified by the following pseudo code. The pseudo code describes only the activity sequencing logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The Start Sequencing Request Process only describes the expected behavior that an implementation will exhibit.

**Start Sequencing Request Behavior.**
1. The candidate activity is the root activity of the activity tree
2. **Loop** – Traverse Activity Tree
   2.1. Apply the Flow Activity Sequencing Subprocess to the candidate activity in the forward direction
   2.2. If the Flow Activity Sequencing Subprocess indicates the candidate activity is not a target for delivery
   Then
   2.2.1. Exit Start Sequencing Request Process – No Delivery Request
   2.3. If the Flow Activity Sequencing Subprocess indicates the candidate activity is a leaf activity and a target for delivery
   Then
   2.3.1. Exit Start Sequencing Request Process – Delivery Request is for the candidate activity
   2.4. Next candidate activity for tree traversal is the Next activity returned by the Flow Activity Sequencing Subprocess

SB.2.8 Continue Sequencing Request Process

A Continue sequencing request is used to identify the next activity to deliver starting at the current activity, traversing forward. The Continue Sequencing Request Process traverses the activity tree, applying sequencing rules and limit conditions to identify the next activity to deliver.

The sequencing process uses the Flow Activity Sequencing Subprocess and the Flow Tree Traversal Subprocess. It does not explicitly use any sequencing definitions or tracking model data.

The sequencing request process is defined as a rule- and limit-checking traversal of the activity tree.

1. The process traverses the entire tree, forward, depth first, from the current activity using the Flow Activity Sequencing Subprocess. At any activity, if only Choice is enabled, stop without identifying an activity to deliver. Check the limit conditions for the activity and stop if any limits are exceeded. Stop if the traversal reaches an activity that is disabled. Skip (and traverse forward) activities marked to be skipped. If the activity is not a leaf, traverse the tree to find the next activity. The first leaf activity found is the candidate activity to deliver.

The process can be described as a tree traversal loop. Start at the current activity, use the Flow Tree Traversal Subprocess to go forward to the next activity. Exit if the activity should not be delivered. Otherwise, traverse the activity tree in a loop. Use the Flow Activity Sequencing Subprocess to determine if the loop should terminate,
because either the activity should or should not be delivered. Otherwise, continue the loop with the next activity in
the tree traversal.

The Continue Sequencing Request Process either validates the sequencing request and returns the candidate activity
to deliver or returns an indication that there is no activity to deliver.

The Continue Sequencing Request Process is specified by the following pseudo code. The pseudo code describes
only the activity sequencing logic. How this process is implemented or how information is encoded, stored,
represented or bound is outside the scope of this specification. The Continue Sequencing Request Process only
describes the expected behavior that an implementation will exhibit.

Continue Sequencing Request Behavior.
1. The candidate activity is the current activity of the activity tree
2. If there is no candidate activity Then
   2.1. Exit Continue Sequencing Request Process – Error, no current activity to continue, No Delivery Request
3. Apply the Flow Tree Traversal Subprocess to the candidate activity in the forward direction
   3.1. If the Flow Tree Traversal Subprocess did not identify the next activity in the tree Then
      3.1.1. Exit Continue Sequencing Request Process – No Delivery Request
   3.2. Else
      3.2.1. Candidate activity is the activity returned by the Flow Tree Traversal Subprocess
4. Loop – Traverse Activity Tree
   4.1. Apply the Flow Activity Sequencing Subprocess to the candidate activity in the forward direction
   4.2. If the Flow Activity Sequencing Subprocess indicates the candidate activity is not a target for delivery
      Then
         4.2.1. Exit Continue Sequencing Request Process – No Delivery Request
   4.3. If the Flow Activity Sequencing Subprocess indicates the candidate activity is a leaf activity and a target for delivery
      Then
         4.3.1. Exit Candidate Sequencing Request Process – Delivery Request is for the candidate activity
   4.4. Next candidate activity for tree traversal is the Next activity returned by the Flow Activity Sequencing Subprocess

SB.2.9 Continue From Parent Sequencing Request Process

A Continue From Parent sequencing request is used to identify the next activity to deliver starting at the parent of
the current activity, traversing forward. The Continue From Parent Sequencing Request Process traverses the
activity tree, applying sequencing rules and limit conditions to identify the next activity to deliver.

The sequencing process uses the Flow Activity Sequencing Subprocess and the Flow Tree Traversal Subprocess. It
does not explicitly use any sequencing definitions or tracking model data.

The sequencing request process is defined as a rule- and limit-checking traversal of the activity tree.

1. The process traverses the entire tree, forward, depth first, from the parent of the current activity using the Flow
   Activity Sequencing Subprocess. At any activity, if only Choice is enabled, stop without indentifying an activity
to deliver. Check the limit conditions for the activity and stop if any limits are exceeded. Stop if the traversal
reaches an activity that is disabled. Skip (and traverse forward) activities marked to be skipped. If the activity
is not a leaf, traverse the tree to find the next activity. The first leaf activity found is the candidate activity to
deliver.

The process can be described as a tree traversal loop. Start at the current activity, use the Flow Tree Traversal
Subprocess to go forward to the next activity. Exit if the activity should not be delivered. Otherwise, traverse the
activity tree in a loop. Use the Flow Activity Sequencing Subprocess to determine if the loop should terminate,
because either the activity should or should not be delivered. Otherwise, continue the loop with the next activity in
the tree traversal.
The **Continue From Parent Sequencing Request Process** either validates the sequencing request and returns the candidate activity to deliver or returns an indication that there is no activity to deliver.

The **Continue From Parent Sequencing Request Process** is specified by the following pseudo code. The pseudo code describes only the activity sequencing logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The **Continue From Parent Sequencing Request Process** only describes the expected behavior that an implementation will exhibit.

**Continue From Parent Sequencing Request Behavior.**

1. The candidate activity is the *parent* of the current activity of the activity tree
2. **If** there is no candidate activity **Then**
   2.1. **Exit** **Continue From Parent Sequencing Request Process** – Error, no current activity to continue, No Delivery Request
3. Apply the **Flow Tree Traversal Subprocess** to the candidate activity in the *forward* direction
   3.1. **If** the **Flow Tree Traversal Subprocess** did not identify the next activity in the tree **Then**
      3.1.1. **Exit** **Continue From Parent Sequencing Request Process** – No Delivery Request
   3.2. **Else**
      3.2.1. Candidate activity is the activity returned by the **Flow Tree Traversal Subprocess**
4. **Loop** – **Traverse Activity Tree**
   4.1. Apply the **Flow Activity Sequencing Subprocess** to the candidate activity in the *forward* direction
   4.2. **If** the **Flow Activity Sequencing Subprocess** indicates the candidate activity is not a target for delivery **Then**
      4.2.1. **Exit** **Continue From Parent Sequencing Request Process** – No Delivery Request
   4.3. **If** the **Flow Activity Sequencing Subprocess** indicates the candidate activity is a leaf activity and a target for delivery **Then**
      4.3.1. **Exit** **Candidate From Parent Sequencing Request Process** – Delivery Request is for the candidate activity
   4.4. Next candidate activity for tree traversal is the *Next* activity returned by the **Flow Activity Sequencing Subprocess**

**SB.2.10 Previous Sequencing Request Process**

A *Previous* sequencing request is used to identify the next activity to deliver starting at the current activity, traversing backward. The **Previous Sequencing Request Process** traverses the activity tree, applying sequencing rules and limit conditions to identify the next activity to deliver.

The sequencing process uses the **Flow Activity Sequencing Subprocess** and the **Flow Tree Traversal Subprocess**. It does not explicitly use any sequencing definitions or tracking model data.

The sequencing request process is defined as a rule- and limit-checking traversal of the activity tree.

1. The process traverses the entire tree, backward, depth first, from the current activity using the **Flow Activity Sequencing Subprocess**. At any activity, if only **Choice** is enabled, stop without indentifying an activity to deliver. Check the limit conditions for the activity and stop if any limits are exceeded. Stop if the traversal reaches an activity that is disabled. Skip (and traverse backward) activities marked to be skipped. If the activity is not a leaf, traverse the tree to find the next (i.e., previous) activity. The first leaf activity found is the candidate activity to deliver.

The process can be described as a tree traversal loop. Start at the current activity, use the **Flow Tree Traversal Subprocess** to go backward to the previous activity. Exit if the activity should not be delivered. Otherwise, traverse the activity tree in a loop. Use the **Flow Activity Sequencing Subprocess** to determine if the loop should terminate, because either the activity should or should not be delivered. Otherwise, continue the loop with the next activity in the tree traversal.
The Previous Sequencing Request Process either validates the sequencing request and returns the candidate activity to deliver or returns an indication that there is no activity to deliver.

The Previous Sequencing Request Process is specified by the following pseudo code. The pseudo code describes only the activity sequencing logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The Previous Sequencing Request Process only describes the expected behavior that an implementation will exhibit.

Previous Sequencing Request Behavior,
1. The candidate activity is the current activity of the activity tree
2. If there is no candidate activity Then
   2.1. Exit Continue Sequencing Request Process – Error, no current activity to continue, No Delivery Request
3. Apply the Flow Tree Traversal Subprocess to the candidate activity in the backward direction
   3.1. If the Flow Tree Traversal Subprocess did not identify the next activity in the tree Then
       3.1.1. Exit Previous Sequencing Request Process – No Delivery Request
   3.2. Else
       3.2.1. Candidate activity is the activity returned by the Flow Tree Traversal Subprocess
4. Loop – Traverse Activity Tree
   4.1. Apply the Flow Activity Sequencing Subprocess to the candidate activity in the backward direction
   4.2. If the Flow Activity Sequencing Subprocess indicates the candidate activity is not a target for delivery Then
       4.2.1. Exit Previous Sequencing Request Process – No Delivery Request
   4.3. If the Flow Activity Sequencing Subprocess indicates the candidate activity is a leaf activity and a target for delivery Then
       4.3.1. Exit Candidate Sequencing Request Process – Delivery Request is for the candidate activity
   4.4. Next candidate activity for tree traversal is the Next activity returned by the Flow Activity Sequencing Subprocess

SB.2.11 Choice Sequencing Request Process

A Choice sequencing request specifies the next activity to deliver. The Choice Sequencing Request Process traverses the activity tree, applying sequencing rules and limit conditions to verify the specified next activity to deliver.

The sequencing process uses the Sequencing Rules Check Subprocess, the Check Activity Subprocess, and the Flow Activity Sequence Subprocess. It does not explicitly use any sequencing definitions or tracking model data.

The sequencing request process is defined as a rule- and limit-checking traversal of the activity tree.

1. The process requires a target activity, i.e., the target of the choice. The process requires a current activity (or the root). Both the parent of the current activity and the parent of the target activity must have Choice enabled. The target must have Hidden From Choice disabled. Traverse the path from the current activity to the target activity and verify that the sequencing rules and limit conditions hold for all activities along the path. There are 5 cases:
   1. The current activity and target are the same: Verify that the target can be delivered
   2. The target activity is a descendant of the current activity (move down the tree): Traverse the tree forward from the current activity to the target activity, applying sequencing rules and limit conditions by using the Choice Activity Sequence Subprocess to validate each activity on the path.
   3. The target activity is an ancestor of the current activity (move up the tree): Traverse the tree backward from the current activity to the target activity, applying sequencing rules and limit conditions by using the Choice Activity Sequence Subprocess to validate each activity on the path.
   4. The target activity is forward in the tree (move up to the common ancestor, down to the target): Traverse the tree backward from the current activity to the common ancestor activity, applying sequencing rules and limit conditions by using the Choice Activity Sequence Subprocess to validate each activity on the path. Traverse the tree forward from the common ancestor activity to the target activity, applying sequencing rules and limit conditions by using the Choice Activity Sequence Subprocess to validate each activity on the path.
rules and limit conditions by using the *Choice Activity Sequence Subprocess* to validate each activity on the path.

5. The target activity is forward in the tree (move up to the common ancestor, down to the target): Traverse the tree backward from the current activity to the common ancestor activity, applying sequencing rules and limit conditions by using the *Choice Activity Sequence Subprocess* to validate each activity on the path. Traverse the tree forward from the common ancestor activity to the target activity, applying sequencing rules and limit conditions by using the *Choice Activity Sequence Subprocess* to validate each activity on the path.

The *Choice Sequencing Request Process* either validates the sequencing request and returns the candidate activity to deliver or returns an indication that there is no activity to deliver.

The *Choice Sequencing Request Process* is specified by the following pseudo code. The pseudo code describes only the activity sequencing logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The *Choice Sequencing Request Process* only describes the expected behavior that an implementation will exhibit.

**Choice Sequencing Request Process.**

1. If there is no target activity Then
   1.1. Exit *Choice Sequencing Request Process* – Error, No choice target activity, No Delivery Request
2. If the target activity is not a leaf activity Then
   2.1. Exit *Choice Sequencing Request Process* – Error, Target is not a leaf activity, No Delivery Request
3. If there are Sequencing Rules with Rule Action attribute value of *Hidden From Choice* for the target activity Then
   3.1. For each such Rule
      3.1.1. Apply the *Sequencing Rules Check Subprocess* to the rule and the activity
      3.1.2. If the *Sequencing Rules Check Subprocess* returns *True* Then
         3.1.2.1. Exit *Choice Sequencing Request Process* – No Delivery Request
4. If there is no current activity Then
   4.1. Use the root of the activity tree as the current activity
5. If there is *parent* for the current activity Then
   5.1. If the Sequencing Control Mode attribute value *Choice* for the *parent* of the current activity is *False* Then
      5.1.1. Exit *Choice Sequencing Request Process* – No Delivery Request
6. If there is *parent* for the target activity Then
   6.1. If the Sequencing Control Mode attribute value *Choice* for the *parent* of the target activity is *False* Then
      6.1.1. Exit *Choice Sequencing Request Process* – No Delivery Request
7. Find the activity that is the common ancestor of the current activity and the target activity
8. **Case:** Current activity and target activity are the same – select the current activity
   8.1. Apply the *Check Activity Subprocess* to the current activity
   8.2. If the *Check Activity Subprocess* returns *True*
      8.2.1. Exit *Choice Sequencing Request Process* – Error, limits or rules violated, No Delivery Request
8.3. Exit *Choice Sequencing Request Process* – Delivery Request is for the target activity
9. **Case:** Current activity and common ancestor activity are the same – path to target is forward in the tree
   9.1. Form the *ActivityPath* as an ordered sequence of all activities from the current activity to the target activity
   9.2. If the *Activity Path* is empty Then
      9.2.1. Exit *Choice Sequencing Request Process* – Error, no activities on path, No Delivery Request
   9.3. For each activity on the *Activity Path*
      9.3.1. Apply the *Choice Activity Sequence Subprocess* to the activity in the forward direction
      9.3.2. If the *Choice Activity Sequence Subprocess* returns *True*
         9.3.2.1. Exit *Choice Sequencing Request Process* – Error, limits or rules violated, No Delivery Request
   9.4. Exit *Choice Sequencing Request Process* – Delivery Request is for the target activity
10. **Case**: Target activity and common ancestor activity are the same – path to the target activity is backward in the tree

10.1. Form the *ActivityPath* as an ordered sequence of all activities from the current activity to the target activity

10.2. **If** the *Activity Path* is empty **Then**

10.2.1. **Exit Choice Sequencing Request Process** – Error, no activities on path, No Delivery Request

10.3. **For** each activity on the *Activity Path*

10.3.1. Apply the *Choice Activity Sequence Subprocess* to the activity in the *backward* direction

10.3.2. **If** the *Choice Activity Sequence Subprocess* returns *True*

10.3.2.1. **Exit Choice Sequencing Request Process** – Error, limits or rules violated, No Delivery Request

10.4. **Exit Choice Sequencing Request Process** – Delivery Request is for the target activity

11. **Case**: Target activity is forward from the current activity – path is forward in the tree

11.1. Form the *ActivityPath* as an ordered sequence of all activities from the current activity to the common ancestor activity

11.2. **If** the *Activity Path* is empty **Then**

11.2.1. **Exit Choice Sequencing Request Process** – Error, no activities on path, No Delivery Request

11.3. **For** each activity on the *Activity Path*

11.3.1. Apply the *Choice Activity Sequence Subprocess* to the activity in the *backward* direction

11.3.2. **If** the *Choice Activity Sequence Subprocess* returns *True*

11.3.2.1. **Exit Choice Sequencing Request Process** – Error, limits or rules violated, No Delivery Request

11.4. Form the *ActivityPath* as an ordered sequence of all activities from the common ancestor activity to the target activity

11.5. **If** the *Activity Path* is empty **Then**

11.5.1. **Exit Choice Sequencing Request Process** – Error, no activities on path, No Delivery Request

11.6. **For** each activity on the *Activity Path*

11.6.1. Apply the *Choice Activity Sequence Subprocess* to the activity in the *forward* direction

11.6.2. **If** the *Choice Activity Sequence Subprocess* returns *True*

11.6.2.1. **Exit Choice Sequencing Request Process** – Error, limits or rules violated, No Delivery Request

11.7. **Exit Choice Sequencing Request Process** – Delivery Request is for the target activity

12. **Case**: Target activity is backward from the current activity – path is backward in the tree

12.1. Form the *ActivityPath* as an ordered sequence of all activities from the current activity to the common ancestor activity

12.2. **If** the *Activity Path* is empty **Then**

12.2.1. **Exit Choice Sequencing Request Process** – Error, no activities on path, No Delivery Request

12.3. **For** each activity on the *Activity Path*

12.3.1. Apply the *Choice Activity Sequence Subprocess* to the activity in the *backward* direction

12.3.2. **If** the *Choice Activity Sequence Subprocess* returns *True*

12.3.2.1. **Exit Choice Sequencing Request Process** – Error, limits or rules violated, No Delivery Request

12.4. Form the *ActivityPath* as an ordered sequence of all activities from the common ancestor activity to the target activity

12.5. **If** the *Activity Path* is empty **Then**

12.5.1. **Exit Choice Sequencing Request Process** – Error, no activities on path, No Delivery Request

12.6. **For** each activity on the *Activity Path*

12.6.1. Apply the *Choice Activity Sequence Subprocess* to the activity in the *forward* direction

12.6.2. **If** the *Choice Activity Sequence Subprocess* returns *True*

12.6.2.1. **Exit Choice Sequencing Request Process** – Error, limits or rules violated, No Delivery Request

12.7. **Exit Choice Sequencing Request Process** – Delivery Request is for the target activity

13. **Exit Choice Sequencing Request Process** – Error, No Delivery Request

**SB.2.12 Retry Sequencing Request Process**
A Retry sequencing request is used to restart the current activity (a new attempt).

The request presumes that the current activity has been terminated. The Navigation Process should have issued an Exit Request to the Exit Process to terminate the activity.

The sequencing process uses the Check Activity Subprocess. It does not explicitly use any sequencing definitions or tracking model data.

The sequencing request process is defined as a rule- and limit-checking process.

1. The Retry Sequencing Request Process checks the status of the current activity, using the Check Activity Subprocess, and simply returns the current activity as the target activity for the delivery request.

The Retry Sequencing Request Process either validates the sequencing request and returns the candidate activity to deliver, returns an indication that there is no activity to deliver, or returns an error.

The Retry Sequencing Request Process is specified by the following pseudo code. The pseudo code describes only the retry sequencing logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The Retry Sequencing Request Process only describes the expected behavior that an implementation will exhibit.

**Retry Sequencing Request Process.**

1. **If** there is no current activity **Then**
   1.1. **Exit** Retry Sequencing Request Process – Error, no current activity to retry, No Delivery Request
2. **If** the current activity is not a leaf activity **Then**
   2.1. **Exit** Retry Sequencing Request Process – Error, current activity is not a leaf activity, No Delivery Request
3. **If** the Activity Status attribute value Activity in Progress for the current activity is True **Then**
   3.1. **Exit** Retry Sequencing Request Process – Error, current activity was not terminated, No Delivery Request
4. Apply the Check Activity Subprocess to the current activity
5. **If** the Check Activity Subprocess returns True
   5.1. **Exit** Retry Sequencing Request Process – Error, limits or rules violated, No Delivery Request
6. **Exit** Retry Sequencing Request Process – Delivery Request is for the current activity

**SB.2.13 Retry All Sequencing Request Process**

A Retry All sequencing request is used to restart the current activity (a new attempt on the activity and all of its parents).

The request presumes that the current activity and all its parent activities have been terminated. The Navigation Process should have issued an Exit Request to the Exit Process to terminate the current activity and its parent activities.

The sequencing process uses the Check Activity Subprocess. It does not explicitly use any sequencing definitions or tracking model data.

The sequencing request process is defined as a rule- and limit-checking process.

1. The Retry All Sequencing Request Process checks the status of the current activity and its parent activities, using the Check Activity Subprocess, and simply returns the current activity as the target activity for the delivery request.

The Retry All Sequencing Request Process either validates the sequencing request and returns the candidate activity to deliver, returns an indication that there is no activity to deliver, or returns an error.
The Retry All Sequencing Request Process is specified by the following pseudo code. The pseudo code describes only the retry sequencing logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The Retry All Sequencing Request Process only describes the expected behavior that an implementation will exhibit.

**Retry All Sequencing Request Process.**

1. If there is no current activity Then
   1.1. Exit Retry All Sequencing Request Process – Error, no current activity to retry, No Delivery Request
2. If the current activity is not a leaf activity Then
   2.1. Exit Retry All Sequencing Request Process – Error, current activity is not a leaf activity, No Delivery Request
3. Form the Activity Path as an ordered sequence of all activities form the current activity to the root of the activity tree
4. If the Activity Path is empty Then
   4.1. Exit Retry All Sequencing Request Process – Error, no activities to retry, No Delivery Request
5. For each activity on the Activity Path
   5.1. If the Activity Status attribute value Activity in Progress for the activity is True Then
      5.1.1. Exit Retry Sequencing Request Process – Error, activity was not terminated, No Delivery Request
      5.1.2. Apply the Check Activity Subprocess to the current activity
      5.1.3. If the Check Activity Subprocess returns True
         5.1.3.1. Exit Retry All Sequencing Request Process – Error, limits or rules violated, No Delivery Request
6. Exit Retry All Sequencing Request Process – Delivery Request is for the current activity

**SB.2.14 Exit Sequencing Request Process**

An Exit sequencing request is used to advance to the next activity in the activity tree on activity exit if the Auto Advance control for the parent of the activity is enabled.

The request presumes the current activity has been terminated. The Navigation Process should have issued an Exit Request to the Exit Process to terminate the activity.

The sequencing process uses Control Modes and information from the activity state model. The sequencing process uses the Flow Tree Traversal Subprocess and the Flow Activity Sequencing Subprocess.

The sequencing request process is defined as a rule- and limit-checking traversal of the activity tree.

1. If the current activity is the root, sequencing is terminated. If the parent of the current activity has either Flow or Choice enabled, sequencing is terminated, no activity is identified for delivery. If the parent of the current activity does not have Auto Advance enabled, sequencing is terminated, no activity is identified for delivery. Otherwise, the process traverses the entire tree, forward, depth first, from the current activity using the Flow Activity Sequencing Subprocess. At any activity, if only Choice is enabled, stop without identifying an activity to deliver. Check the limit conditions for the activity and stop if any limits are exceeded. Stop if the traversal reaches an activity that is disabled. Skip (and traverse forward) activities marked to be skipped. If the activity is not a leaf, traverse the tree to find the next activity. The first leaf activity found is the candidate activity to deliver.

The Exit Sequencing Request Process either validates the sequencing request and returns the candidate activity to deliver, returns an indication that the there is no activity to deliver, or returns an error.

The Exit Sequencing Request Process is specified by the following pseudo code. The pseudo code describes only the exit sequencing logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The Exit Sequencing Request Process only describes the expected behavior that an implementation will exhibit.
Exit Sequencing Request Process.
1. **If** there is no current activity **Then**
   1.1. **Exit Exit Sequencing Request Process** – Already exited, No Delivery Request
2. **If** the Activity Status attribute value **Activity in Progress** for the current activity is **True** **Then**
   2.1. **Exit Exit Sequencing Request Process** – Error, current activity was not terminated, No Delivery Request
3. **If** the current activity is the root of the activity tree **Then**
   3.1. **Exit Exit Sequencing Request Process** – No Delivery Request, Exit Sequencing
4. **If** there is no parent for the current activity **Then**
   4.1. **Exit Exit Sequencing Request Process** – Error, No parent to apply request to, No Delivery Request
5. **If** the Sequencing Control Mode attribute value **Flow** for the parent of the current activity is **True** **Then**
   5.1. **Exit Exit Sequencing Request Process** – No Delivery Request
6. **If** the Sequencing Control Mode attribute value **Choice** for the parent of the current activity is **True** **Then**
   6.1. **Exit Exit Sequencing Request Process** – No Delivery Request
7. **If** the Sequencing Control Mode attribute value **Auto Advance** for the parent of the current activity is **False** **Then**
   7.1. **Exit Exit Sequencing Request Process** – No Delivery Request
8. The candidate activity is the current activity of the activity tree
9. Apply the **Flow Tree Traversal Subprocess** in the forward direction from the candidate activity
   9.1. **If** the **Flow Tree Traversal Subprocess** did not identify the next activity in the tree **Then**
      9.1.1. **Exit Exit Sequencing Request Process** – No Delivery Request
   9.2. **Else**
      9.2.1. Candidate activity is the activity returned by the **Flow Tree Traversal Subprocess**
10. **Loop** – **Traverse Activity Tree**
    10.1. Apply the **Flow Activity Sequencing Subprocess** to the candidate activity in the forward direction
    10.2. **If** the **Flow Activity Sequencing Subprocess** indicates the candidate activity is not a target for delivery **Then**
        10.2.1. **Exit Exit Sequencing Request Process** – No Delivery Request
    10.3. **If** the **Flow Activity Sequencing Subprocess** indicates the candidate activity is a leaf activity and a target for delivery **Then**
        10.3.1. **Exit Exit Sequencing Request Process** – Delivery Request is for the candidate activity
    10.4. Next candidate activity for tree traversal is the Next activity returned by the **Flow Activity Sequencing Subprocess**

SB.2.15 Exit Parent Sequencing Request Process

An **Exit Parent** sequencing request is used to stop the sequencing process if the parent of the current activity is the root of the activity tree.

The request presumes the current activity and its parent activity have been terminated. The **Navigation Process** should have issued an **Exit Request** to the **Exit Process** to terminate the current activity and its parent.

The sequencing process uses information from the activity state model.

The sequencing request process is defined as a rule- and limit-checking process.

1. The **Exit Parent Sequencing Request Process** checks the status of the current activity and its parent activity and does not return an activity for delivery.

The **Exit Parent Sequencing Request Process** either validates the sequencing request and returns an indication that sequencing is to terminate, or returns an error.

The **Exit Parent Sequencing Request Process** is specified by the following pseudo code. The pseudo code describes only the exit sequencing logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The **Exit Parent Sequencing Request Process** only describes the expected behavior that an implementation will exhibit.
Exit Parent Sequencing Request Process.

1. If there is no current activity Then
   1.1. Exit Exit Parent Sequencing Request Process – Already exited, No Delivery Request
2. If there is no parent for the current activity Then
   2.1. Exit Exit Parent Sequencing Request Process – Error, No parent to apply request to, No Delivery Request
3. If the Activity Status attribute value Activity in Progress for the current activity is True Then
   3.1. Exit Exit Parent Sequencing Request Process – Error, current activity was not terminated, No Delivery Request
4. If the Activity Status attribute value Activity in Progress for the parent of the current activity is True Then
   4.1. Exit Exit Parent Sequencing Request Process – Error, parent of current activity was not terminated, No Delivery Request
5. If the parent of the current activity is the root of the activity tree Then
   5.1. Exit Exit Parent Sequencing Request Process – No Delivery Request, Exit Sequencing
6. Exit Exit Parent Sequencing Request Process – No Delivery Request

SB.2.16 Exit All Sequencing Request Process

An Exit All sequencing request is used to stop the sequencing process.

The request presumes the current activity and all its parent activities have been terminated. The Navigation Process should have issued an Exit Request to the Exit Process to terminate the activity and its parent activities.

The sequencing process uses information from the activity state model.

The sequencing request process is defined as a rule- and limit-checking process.

1. The Exit All Sequencing Request Process checks the status of the current activity and its parent activities and does not return an activity for delivery. The current activity becomes undefined.

The Exit All Sequencing Request Process either validates the sequencing request and returns an indication that sequencing is to terminate, or returns an error.

The Exit All Sequencing Request Process is specified by the following pseudo code. The pseudo code describes only the exit sequencing logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The Exit All Sequencing Request Process only describes the expected behavior that an implementation will exhibit.

Exit All Sequencing Request Process.

1. If there is no current activity Then
   1.1. Exit Exit All Sequencing Request Process – Already exited, No Delivery Request
2. Form the ActivityPath as an ordered sequence of all activities from the current activity to the root of the activity tree
3. If the Activity Path is empty Then
   3.1. Exit Exit All Sequencing Request Process – Error, no activities to exit, No Delivery Request
4. For each activity on the Activity Path
   4.1. If the Activity Status attribute value Activity in Progress for the activity is True Then
       4.1.1. Exit Exit Sequencing Request Process – Error, activity was not terminated, No Delivery Request
5. Set the Current Activity to undefined
6. Exit Exit All Sequencing Request Process – No Delivery Request, Exit Sequencing

SB.2.17 Sequencing Request Process
The **Sequencing Request Process** is the overall control process that handles all sequencing requests. It simply dispatches a request to an appropriate processor. The individual request processors will do one of the following:

- Indicate that the sequencing process does not identify a candidate activity to deliver:
  - These is no candidate – the overall sequencing system will wait for the next navigation request.
  - There is a candidate, but the limit conditions are exceeded – The overall sequencing system will wait for the next navigation request.
- Indicate that the entire sequencing process should terminate.
- Return candidate activity to deliver.
- Return an error.

The **Sequencing Request Process** for a sequencing request is specified by the following pseudo code. The pseudo code describes only the request and tree traversal logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The **Sequencing Request Process** only describes the expected behavior that an implementation will exhibit.

### Sequencing Request Behavior.

1. **Case:** Sequencing Request is **Start**
   1.1. Apply the **Start Sequencing Request Process**
   1.2. Return the Delivery Request or Error
2. **Case:** Sequencing Request is **Continue**
   2.1. Apply the **Continue Sequencing Request Process**
   2.2. Return the Delivery Request or Error
3. **Case:** Sequencing Request is **Continue From Parent**
   3.1. Apply the **Continue From Parent Sequencing Request Process**
   3.2. Return the Delivery Request or Error
4. **Case:** Sequencing Request is **Previous**
   4.1. Apply the **Previous Sequencing Request Process**
   4.2. Return the Delivery Request or Error
5. **Case:** Sequencing Request is **Choice**
   5.1. Apply the **Choice Sequencing Request Process**
   5.2. Return the Delivery Request or Error
6. **Case:** Sequencing Request is **Retry**
   6.1. Apply the **Retry Sequencing Request Process**
   6.2. Return the Delivery Request or Error
7. **Case:** Sequencing Request is **Retry All**
   7.1. Apply the **Retry All Sequencing Request Process**
   7.2. Return the Delivery Request or Error
8. **Case:** Sequencing Request is **Exit**
   8.1. Apply the **Exit Sequencing Request Process**
   8.2. Return the Delivery Request, Exit Request or Error
9. **Case:** Sequencing Request is **Exit Parent**
   9.1. Apply the **Exit Parent Sequencing Request Process**
   9.2. Return the Delivery Request, Exit Request or Error
10. **Case:** Sequencing Request is **Exit All**
   10.1. Apply the **Exit All Sequencing Request Process**
   10.2. Return the Delivery Request, Exit Request or Error

### SB.3 Notes/Changes from IMS SS Public Draft

This model is identical in functionality and intent with that described in the IMS SS Public Draft. Descriptions and terminology have been simplified. Inconsistencies and partially specified behaviors have been fully specified.

References to SS data, application profile specific–behaviors and other descriptions not pertaining to just the sequencing behavior have been eliminated. This part just describes the sequencing behavior itself, not how the sequencing process is used within the overall sequencing framework.
The following behaviors have been changed or simplified:

- Various sequencing requests were defined in terms of an *Auto Advance* control behavior. In the data model, auto advance is described as a way to generate a continue request on activity exit. It is not a control that alters how a sequencing request is processed. Auto advance was dropped from the description of the request behavior.

- The statement in the Public Draft “*Choice* control enabled for the children” was interpreted to mean that the Choice control mode was set for the activity, and thus it applies to the children. It was not interpreted to mean that the children of the activity had the choice control mode that then applied to their children.

- Sequencing request behaviors were defined in the Public Draft to include activity termination. Sequencing only identifies activities. Exit is performed externally (and prior) to sequence request processing.

- The description that exit and retry requests ignore sequencing control modes has been dropped. The defined behavior is already independent of control mode.

- The Exit request in the Public Draft defined the behavior as a wait if “*Flow* and/or *Choice* control modes are defined …”. “and/or” was interpreted as “or”.

- *Hidden from Choice* is applied to only the target activity of Choice. It is not applied to activities in the tree traversal.

- If the sequencing process cannot identify something to deliver, the overall sequencing system waits. The next event is a navigation request, not a sequencing request.

**SB.4 Open Issues**

The sequencing process does not describe how to deal with activity threads. It makes sense to process the sequencing request for the entire activity path for the main thread. How to process other threads is not obvious – should the processing be applied to the entire activity thread, skipping only those activities that are not part of the thread, or should it be applied to a different path? [SS.SB.01]

Should the thread be specified, or taken from the thread assigned to the activity specified in the sequencing request? [SS.SB.02]

The Retry request is defined to ignore the control modes of the activity. Control modes are applied for the children of the activity. Ignoring control modes does not seem to make sense, especially since the activity will be a leaf. [SS.SB.03]

The Retry All request is defined to ignore the control modes of the cluster, i.e., the parent of the activity. Is this correct, or is it the grandparent, so the request applies to the parent? [SS.SB.04]

The Retry All request identifies only the activity to deliver. Delivery presents an activity, not a path. [SS.SB.05]

It’s not clear if all the combinations of *Continue* and *Exit* with *Flow*, *Choice* and *Auto Advance* are represented, or if there is a simpler way to express all of the combinations. (See Section SB.8) [SS.SB.06]

*Exit* with *Flow* and *Auto Advance* results in a wait for another request. Should the result be *Continue/Auto Advance* if *Flow* is enabled with *Auto Advance?* [SS.SB.07]
What does it mean to say that “sequencing requests are permitted regardless of control mode of the cluster”? Sequencing requests are always permitted. Is this a meaningless statement, or does it mean the the control modes of the cluster are ignored? [SS.SB.08]

Various sequencing requests are defined to continue until reaching a leaf. This assumes there is no content to deliver at any node except the leaf. Is this assumption explicitly stated anywhere? [SS.SB.09]

Should limit conditions be checked when first visiting an activity and before applying rules, or should limit conditions only be checked when an activity is identified as a candidate or when traversing to another activity? Does it matter? [SS.SB.10]

Does Previous require that Forward Only be False for only the current activity, or that it be False for all activities on the backward path? [SS.SB.11]

Exit and Continue from Parent were not defined if the current activity was the root (no parent). The assumed behavior is that if there is no parent, the action is an error. [SS.SB.12]

In the data model, rules are defined to be unordered. The behavior groups rules, but treats the rules within the group as unordered. This provides a hierarchy, e.g., any disable rule will override all other rules. All precondition rules are not equal, and all rules are not applicable in all sequencing request cases, so simple evaluation of all rules is not appropriate. Should rules be grouped or evaluated in some other order? The current behavior in flow traversal could be replaced by a case statement that says all precondition rules should be considered until one fires. [SS.SB.13]

In a backward flow walk, limit conditions are evaluated both when moving up and down the tree. Should they only be evaluated when going down the tree? [SS.SB.14]

The terminology of “exit sequencing” may be confused with “exit activity” or “exit process”. The exit sequencing request is an activity exit. The exit all request is a process exit. Alternative terminology should be used to clarify the distinction. [SS.SB.15]

Choice requires that both the current activity and the target have choice enabled. It is logical that the target must have choice enabled. Requiring choice on the current activity is essentially a statement that “you can leave the cluster”. Overloading choice with two meanings, one for when you can exit and one for when you can enter a cluster is a bad design. There should be two separate controls: choice and choice exit, and the choice request should be dependent only on the choice exit mode of the current activity (default true). [SS.SB.16]

Choice is defined only if there is a parent of the current activity. This does not handle the case of only a root activity. The behavior provided treats choice of root as always permitted. [SS.SB.17]

In Choice, if the target is backward, traversal from the current activity to the common ancestor is backward. Is the traversal from the common ancestor to the target also backward or is it forward since it is downward in the tree? [SS.SB.18]

In Choice, should traversal stop if it reaches an activity that is disabled? [SS.SB.19]

The Choice Activity Sequence Subprocess ignores Forward Only conditions if the activity does not have a parent. [SS.SB.20]

The treatment of activities that do not have a parent may not be consistent. [SS.SB.21]

**SB.5 Implementation Recommendations**

Values of Completion and Measure are normalized between 0..1. Testing for the exact value of 1.0 is unreliable. Any value > 0.99999 should be considered equal to 1.0 in evaluating rule conditions (based on the required precision of 5 decimal digits).
Normalized data values are described as value space real. A decimal data type should be considered as an alternative to a float data type for the implementation to maintain the required precision.

Rule processors should use short circuit evaluation techniques when determining if the rule conditions are true. For example, assume an And rule contains a condition on both Objective Data Status and Objective Satisfied Status. If the Objective Data Status evaluates to False, the condition on the the value of Objective Satisfied Status should not be evaluated (in this case, the Objective Satisfied Status value is unreliable). The rule processor may reorder conditions to enable short circuit evaluation. However, short circuit evaluation is not required, as the default values insure that the same sequencing behaviors will result whether short circuit evaluation is not used or not.

The implementation must ensure that data values are consistent when being accessed, i.e., the values of the set of activities that participate in a child set must be consistent while the entire set of rules applied during sequencing are evaluated. Implementing proper transactional controls are left to the implementation.

Processes use data values from the sequencing definition model. Each attribute of the model has a value or inherits the value of the parent activity as specified in the sequencing definition model. In addition, there is a defined default value for each attribute. Processes that access data must access the proper value, i.e., the value of the attribute for the activity or the value for the parent if the data does not exist and is inherited. The implementation may fully instantiate the activity tree with values or provide an inheritance mechanism, either in the data model itself or as part of the sequencing process. The choice of approach is left to the implementation, which must only provide the proper data for use in the sequencing process.

There are no constraints on the order in which the limit conditions or precondition sequencing rules are applied.

There is no requirement that the constraint checking processes identify that multiple limits and preconditions are violated. There is no restriction that the constraint checking processes stop after finding the first constraint that is violated.

**SB.6 Bindings**

Not applicable to a behavior model.

**SB.7 Sequencing Process Service**

The sequencing process may be implemented as a service. The sequencing process has one service access point (SAP) for the Sequencing Request Process.

**SB.7.1 Sequencing Request Process SAP**

**Request:**
The Sequencing Request Process requires data items to specify:
- The designation of the activity tree.
- The root activity of the tree for sequencing. An Xpath expression specifying the root node as a part of a larger tree. Specifying a root permits sequencing to be applied to part of a larger tree structure. The root for sequencing may be the same as the root of the activity tree.
- The sequencing request (from the Sequencing Request vocabulary).
- The target sequencing request. An optional Xpath expression specifying a node within the activity tree. Applicable only for “choice” requests.
- The last activity identified for sequencing. An Xpath expression specifying a node within the activity tree. Optional (will be ignored for some types of sequencing requests).
- The sequencing thread. An integer.
- The learner.
Process:
The *Sequencing Request Process* is defined above (section SB.2.17).

Results:
The *Sequencing Request Process* returns the following upon completion:
- A Boolean indication if the *Sequencing Request Process* was successful.
- An optional indication of the activity to deliver if the process was successful. An Xpath expression specifying a node within the activity tree.
- An optional indication to Exit Sequencing.
- An error indication if the process was not successful. Possible errors include:
  - Data errors:
    - The activity tree was not properly specified.
    - The root of the tree was not properly specified.
    - The sequencing request was not properly specified.
    - The last activity identified for sequencing was not properly specified.
    - The thread was not properly specified (e.g., not 1, or there was no path from the root to the end activity for the specified thread).
    - The learner was not properly specified.
  - Processing Errors:
    - Error in the *Sequencing Request Process*.

The process description does not currently define how to return multiple error indicators. The process description does not detail errors from any of the other processes or subprocesses.

Related Services:
The *Sequencing Request Process* relies on the following other services:
- Time service (return current time point).

### SB.8 Sequencing Requests / Sequencing Control Mode Combinations

There are 160 or $10 \times 2^4$ combinations of *Sequencing Requests* and *Sequencing Control Modes*. All must be specified.

<table>
<thead>
<tr>
<th>Sequencing Request</th>
<th>Choice</th>
<th>Flow</th>
<th>Advance</th>
<th>Forward</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start</strong></td>
<td>N N N N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N N N N Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N N Y N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N N Y Y Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N Y N N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N Y Y N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y N N N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y N N Y Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y N Y N Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y Y N N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y Y Y N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y Y Y Y Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Continue</strong></td>
<td>N N N N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N N N N Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N N Y N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N N Y Y Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Choice</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Retry</strong></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Retry All</strong></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Exit</strong></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>-------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Exit Parent</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Exit All</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
DB. Delivery Behavior Model (Just Delivery V0.7)

The SS process identifies an activity whose content resources are to be delivered to the learner (a delivery request). The identified content resources should then be delivered to the learner. Before the content is actually delivered, the SS process must validate that the content resources from the identified activity may be delivered, i.e., all of the conditions that apply to the delivery of the content for the activity and attempt still hold. The process of validating and then initiating the content delivery is called the “delivery process”.

The delivery process makes no assumptions as to how or when an activity for delivery is identified, including if or when the delivery request resulted from a sequencing request. The delivery process must insure that delivery requests are valid. Once the delivery request is validated, the content resources for the activity are delivered. The delivery process must ensure that certain elements of the tracking model and the activity state model are updated as part of content delivery.

The mechanisms used to actually deliver the content resources for the activity to the learner are not specified. The mechanisms used to update the tracking data for objectives and activities for the learner as the result of delivering the content resources for the activity are not specified.

The overall sequencing process (see Overall Sequencing Process Document OP) relates the delivery process to the navigation, sequencing, exit, and rollup processes.

The delivery process is controlled by parts of the sequencing definition model:
- *Sequencing Rule Definitions* – rules that indicate preconditions on when the content resources for the activity may be delivered.
- *Limit Conditions* – limits on when the content resources for an activity may be delivered.
- *Delivery Controls* – actions applied when or while the content resources for the activity are delivered.

The delivery process uses all parts of the tracking model:
- *Objective Information* – information about the results of the learner’s interactions related to an objective.
- *Progress Information* – information about a learner’s attempt at an activity, updated by the delivery process.

The delivery process uses data from the activity state model.

The behavior of the delivery process is defined in terms of different processes, each of which is applied to one part of the sequencing definition model:
- *Sequencing Rules Process* – applies certain precondition sequencing rules to determine if the delivery request is valid.
- *Limit Conditions Process* – applies limit conditions to determine if the delivery request is valid.

These individual processes are part of the overall process that controls all delivery behavior. The actual delivery is governed by a set of delivery controls and tracking requirements. Subsequent to delivery, the content operates in a managed delivery environment.

**DB.1 Delivery Behavior**

The delivery behavior describes how a sequencing system interprets a delivery request in combination with the elements of the sequencing definition model and with instance data from the tracking model to validate the delivery request and initiate the delivery of the content resources for the activity and tracking of the activity.
Delivery requests are validated using only certain sequencing rules and limit conditions. No other limits, rules, conditions or control modes are applied when validating a delivery request.

An implementation must be capable of representing the processes described and have the implemented process exhibit the behavior described. There are no additional requirements on implementing the delivery behavior model.

The delivery behavior relies on the data descriptions from the sequencing definition model (see Sequencing Definition Model Document SM) and the tracking model (see Tracking Model Document TM). These data models also specify default data values or data inheritance in the activity tree that govern the access to activity or tracking data.

**DB.1.1 Sequencing Rules Process**

The *Sequencing Rules Process* applies some of the precondition sequencing rule descriptions from the sequencing definition model to determine if the content resources for the identified activity may be delivered.

Sequencing rule preconditions are specified by the Sequencing Rule Definitions. The only applicable precondition sequencing rule is the “disable” rule, i.e., the activity is disabled and cannot be delivered (the other preconditions are applicable only when selecting activities). Evaluating the sequencing rule uses data values for Objective and Activity Progress attributes of the tracking model and Limit Condition attributes.

The *Sequencing Rules Process* either validates the delivery request or returns an indication that the delivery request is invalid.

Sequencing rules are always applied in combination with limit conditions. All applicable rules and limit conditions must be True to have a valid content delivery request.

The *Sequencing Rules Process* for an activity is specified by the following pseudo code. The pseudo code describes only the rule checking logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The *Sequencing Rules Process* only describes the expected behavior that an implementation will exhibit.

**Sequencing Rule Behavior.**

1. **If** there are Sequencing Rules for the activity where the attribute value *Rule Action is Disabled* Then
   1.1. **For** each Sequencing Rule for the activity where the attribute value *Rule Action is Disabled*
      1.1.1. Initialize *Activity Condition Bag* as an empty collection
      1.1.2. **For** each *Rule Condition* for the Sequencing Rule for the activity
         1.1.2.1. Evaluate the *Rule Condition* for the rule by applying the Sequencing Rule attribute value
                  *Rule Condition* for the activity to the appropriate corresponding data (either a value from the
                  objective associated with the activity, a progress value for the activity, or a Limit Condition
                  attribute value for the activity)
         1.1.2.2. **If** the *Rule Condition Operator* for the rule is *Not* Then
                  1.1.2.2.1. Negate the Rule Condition evaluation result
         1.1.2.3. **If** the Rule Condition for the rule evaluates to *True* Then
                  1.1.2.3.1. Add a *True* value to the *Activity Condition Bag*
         1.1.2.4. **If** the Rule Condition for the rule evaluates to *False* Then
                  1.1.2.4.1. Add a *False* value to the *Activity Condition Bag*
      1.1.3. **If** the *Activity Children Status Bag* is empty Then
         1.1.3.1. **Exit Sequencing Rules Process** – Delivery request valid
      1.1.4. **Apply** the combination (*And, Any*) from the Sequencing Rule attribute value *Rule Combination* for the activity to the *Activity Condition Bag* to produce a single combined result for the Rule Conditions of the rule of the activity
      1.1.5. **If** the combination condition evaluates to *True* Then
         1.1.5.1. **Exit Sequencing Rules Process** – Delivery request not valid: Activity Disabled
      1.1.6. **If** the combination condition evaluates to *False* Then
1.1.6.1. **Exit Sequencing Rules Process** – Delivery request valid

2. **Exit Sequencing Rules Process** – Delivery request valid

**DB.1.2 Limit Conditions Process**

The *Limit Conditions Process* applies the limit conditions from the sequencing definition model to determine if the content resources from the identified activity may be delivered. If the limit conditions are met or exceeded, the activity identified in the delivery request should not be delivered.

Limit conditions are specified by all of the Limit Condition attributes in the sequencing definition model. These attributes constrain the number of access attempts, the duration of access or time limit on access and the time frame of access. Limit conditions are evaluated using data values for Activity Progress attributes of the tracking model.

The *Limit Conditions Process* either validates the delivery request or returns an indication of the limit(s) that were exceeded causing the delivery request to be invalid.

Limit conditions are always applied in combination with precondition sequencing rules. All applicable rules and limit conditions must be true to have a valid content delivery request.

The *Limit Conditions Process* for an activity is specified by the following pseudo code. The pseudo code describes only the limit condition logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The *Limit Conditions Process* only describes the expected behavior that an implementation will exhibit.

**Limit Conditions Behavior.**

1. **If** the Limit Condition attribute value *Limit Condition Attempt Control* for the activity is *True* **Then**
   1.1. **If** the Activity Progress attribute value *Activity Attempt Count* for the activity is greater than or equal to the Limit Condition attribute value *Limit Condition Attempt Limit* for the activity **Then**
      1.1.1. **Exit Limit Conditions Process** – Delivery request not valid: Activity Limit Exceeded
   2. **If** the Limit Condition attribute value *Limit Condition Activity WallClock Duration Control* for the activity is *True* **Then**
      2.1. **If** the Activity Progress attribute value *Activity WallClock Duration* for the activity is greater than or equal to the Limit Condition attribute value *Limit Condition Activity WallClock Duration Limit* for the activity **Then**
         2.1.1. **Exit Limit Conditions Process** – Delivery request not valid: Activity WallClock Duration Limit Exceeded
   3. **If** the Limit Condition attribute value *Limit Condition Activity Interaction Duration Control* for the activity is *True* **Then**
      3.1. **If** the Activity Progress attribute value *Activity Interaction Duration* for the activity is greater than or equal to the Limit Condition attribute value *Limit Condition Activity Interaction Duration Limit* for the activity **Then**
         3.1.1. **Exit Limit Conditions Process** – Delivery request not valid: Activity Interaction Duration Limit Exceeded
   4. **If** the Limit Condition attribute value *Limit Condition Activity Attempt WallClock Duration Control* for the activity is *True* **Then**
      4.1. **If** the Activity Progress attribute value *Activity Attempt WallClock Duration* for the activity attempt is greater than or equal to the Limit Condition attribute value *Limit Condition Activity Attempt WallClock Duration Limit* for the activity attempt **Then**
         4.1.1. **Exit Limit Conditions Process** – Delivery request not valid: Activity Attempt WallClock Duration Limit Exceeded
   5. **If** the Limit Condition attribute value *Limit Condition Activity Attempt Interaction Duration Control* for the activity is *True* **Then**
      5.1. **If** the Activity Progress attribute value *Activity Attempt Interaction Duration* for the activity attempt is greater than or equal to the Limit Condition attribute value *Limit Condition Activity Attempt Interaction Duration Limit* for the activity **Then**
5.1.1. **Exit Limit Conditions Process** – Delivery request not valid: Activity Attempt Interaction Duration Limit Exceeded

6. **If** the Limit Condition attribute value *Limit Condition Time Limit Control* for the activity is True **Then**
   
6.1. **If** the current timepoint is before the Limit Condition attribute value *Limit Conditions Begin Time Limit* for the activity **Then**
   
6.1.1. **Exit Limit Conditions Process** – Delivery request not valid: Activity Begin Time Limit not met

6.2. **If** the current timepoint is after the Limit Control attribute value *Limit Conditions End Time Limit* for the activity **Then**
   
6.2.1. **Exit Limit Conditions Process** – Delivery request not valid: Activity End Time Limit not met

7. **Exit Limit Conditions Process** – Delivery request valid

**DB.1.3 Overall Delivery Process**

The *Overall Delivery Process* is used to validate the identified activity before delivery. The process is applied to:

- An activity tree, designated by the root activity of the tree.
- The activity within the tree that is identified for delivery.
- The sequencing thread.
- An indication of the learner.

The root of the tree and the activity within the tree specify a unique single path through the tree, designated the delivery path. The delivery validation checks are applied to all activities along that path, starting from the root activity to the activity identified for delivery.

At each step along the delivery path, all delivery processes, *Sequencing Rules Process*, and *Limit Conditions Process* are applied. When the *Overall Delivery Process* is initiated, it is stopped when the delivery request cannot be validated or when the processing successfully reaches the end of the delivery path. If the end of the path is reached, the content resources may be delivered.

The *Overall Delivery Process* is specified by the following pseudo code. The pseudo code describes only the delivery validation logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The *Overall Delivery Process* only describes the expected behavior that an implementation will exhibit.

**Overall Delivery Process.**

1. Form the *Activity Path* as an ordered sequence of all activities from the root of the activity tree the activity to the activity specified in the delivery request
2. **If** the *Activity Path* is empty **Then**
   2.1. **Exit Overall Delivery Process** – Delivery request not valid
3. **For** each activity in the *Activity Path*
   3.1. Apply the *Sequencing Rules Process* to the activity
      3.1.1. **If** the process indicates that the delivery request is not valid **Then**
      3.1.1.1. **Exit Overall Delivery Process** – Delivery request not valid
   3.2. Apply the *Limit Conditions Process* to the activity
      3.2.1. **If** the process indicates that the delivery request is not valid **Then**
      3.2.1.1. **Exit Overall Delivery Process** – Delivery request not valid
4. **Exit Overall Delivery Process** – Delivery request is valid

**DB.2 Content Delivery Environment**

Once the delivery request has been validated, the content is delivered. The delivery environment must deliver the content resources for the activity to the learner. The process of delivering the content, e.g., retrieving it from a content repository, delivering it via an HTTP response, is not specified.
The *Content Delivery Environment Process* controls how the activity is started or attempted and how data for the activity is recorded.

If the activity is *Tracked*, then tracking and status data should be recorded. Tracking and status data should not change if the activity is not tracked.

If the activity was suspended, the activity is resumed when the activity is delivered; otherwise, the delivery is a new attempt.

The *Content Delivery Environment Process* for an activity is specified by the following pseudo code. The pseudo code describes only what should happen once the content has been delivered to the learner. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The *Content Delivery Environment Process* only describes the expected behavior that an implementation will exhibit.

**Content Delivery Environment Process.**

1. **If** the Delivery Controls attribute value *Tracked* for the activity is *False* **Then**
   1.1. The Objective and Activity Progress data for the activity attempt should not be recorded during delivery

2. Once the delivery of the content resources for the activity begins
   2.1. Form the *Activity Path* as an ordered sequence of all activities from the root of the activity tree the activity being delivered
   2.2. **For each activity in the Activity Path**
      2.2.1. **If** the Activity Status attribute value *Activity Suspended* for the activity is *True* **Then**
       2.2.1.1. **If** the Activity Status attribute value *Activity Suspended* for the activity is *True* **Then**
        2.2.1.1.1. Set the Activity Status attribute value *Activity Suspended* for the activity to *False*
       2.2.1.2. **Else**
       2.2.1.2.1. The Activity Progress attribute value *Activity Attempt Count* for the activity is incremented

   2.2.2. **Set the Activity Status attribute value Activity in Progress for the activity to True**
   2.2.3. The delivery environment begins tracking the *Activity Attempt WallClock Duration* and the *Attempt Interaction Duration*

How the delivery environment tracks data, including time durations, is not specified. The delivery environment may be able to determine if the content delivery was attempted, but may not be able to determine if the content was actually delivered to the learner (e.g., an HTTP server responds to the request for content but there may be no mechanism to determine if the client browser received and displayed the content).

The delivery environment is also responsible for recording any other necessary data to support tracking and navigation requests.

**DB.3 Notes/Changes from IMS SS Public Draft**

This model is identical in functionality and intent with that described in the IMS SS Public Draft. Descriptions and terminology have been simplified. Inconsistencies and partially specified behaviors have been fully specified.

References to SS data, application profile–specific behaviors and other descriptions not pertaining to just the delivery behavior have been eliminated. This part just describes the delivery behavior itself, not how the delivery process is used by the sequencing process.

**DB.4 Open Issues**
The Overall Delivery Process does not describe how to deal with activity threads. It makes sense to validate the delivery request for the entire activity path for the main thread. How to validate other threads is not obvious – should the validation be applied to the entire activity thread, skipping only those activities that are not part of the thread, or should it be applied to a different path? [SS.DB.01]

Should the thread be specified, or taken from the thread assigned to the activity specified in the delivery request? [SS.DB.02]

The details of what happens during content delivery may not be complete. Are all the other requirements defined in the other sequencing processes, or are there other requirements that are need to be included here? [SS.DB.03]

The Overall Delivery Process does not include content repository lookup, content handle resolution, launch, etc. While these are out of scope for SS, the overall processing model must indicate how and when the occur. As defined, delivery is not part of the content environment process but could be. [SS.DB.04]

The Overall Delivery Process does not detail what happens if the identified activity for delivery does not have an associated content resource for delivery. [SS.DB.05]

The Overall Delivery Process assumes attempt count data can be incremented after the activity has been delivered. Some delivery environments, e.g., those that use the CMI API, should wait until they verify that the content has been delivered before recording attempt count data. Controlling when the attempt count data (or other tracking data) is set could be controlled by additional delivery attributes that differentiate how content behaves. [SS.DB.06]

**DB.5 Implementation Recommendations**

An implementation may short circuit the validation of a delivery request if both the sequencing and delivery processes are part of a single monolithic process. Such a short circuit elimination of the validation of the delivery request should only be used if the system can insure that none of tracking data used in the rule condition or limit conditions has changed from when the delivery request was generated to when the request for delivery was actually initiated.

Values of Completion and Measure are normalized between 0..1. Testing for the exact value of 1.0 is unreliable. Any value > 0.99999 should be considered equal to 1.0 in evaluating rule conditions (based on the required precision of 5 decimal digits).

Normalized data values are described as value space real. A decimal data type should be considered as an alternative to a float data type for the implementation to maintain the required precision.

Rule processors should use short circuit evaluation techniques when determining if the rule conditions are true. For example, assume an And rule contains a condition on both Objective Data Status and Objective Satisfied Status. If the Objective Data Status evaluates to False, the condition on the the value of Objective Satisfied Status should not be evaluated (in this case, the Objective Status Status value is unreliable). The rule processor may reorder conditions to enable short circuit evaluation. However, short circuit evaluation is not required, as the default values insure that the same sequencing behaviors will result whether short circuit evaluation is not used or not.

The implementation must ensure that data values are consistent when being accessed, i.e., the values of the set of activities that participate in a child set must be consistent while the entire set of rules applied during delivery are evaluated. Implementing proper transactional controls are left to the implementation.

Processes use data values from the sequencing definition model. Each attribute of the model has a value or inherits the value of the parent activity as specified in the sequencing definition model. In addition, there is a defined default value for each attribute. Processes that access data must access the proper value, i.e., the value of the attribute for the activity or the value for the parent if the data does not exist and is inherited. The implementation may fully instantiate the activity tree with values or provide an inheritance mechanism, either in the data model itself or as part
of the delivery process. The choice of approach is left to the implementation, which must only provide the proper
data for use in the delivery process.

There are no constraints on the order in which the limit conditions or precondition sequencing rules are applied.

There is no requirement that the constraint checking processes identify that multiple limits and preconditions are
violated. There is no restriction that the constraint checking processes stop after finding the first constraint that is
violated.

**DB.6 Bindings**

Not applicable to a behavior model.

**DB.7 Delivery Process Service**

The delivery process may be implemented as a service. The delivery process has one service access point (SAP) for
the *Overall Delivery Process* and one SAP for the *Content Delivery Environment Process*.

**DB.7.1 Overall Delivery Process SAP**

**Request:**

The *Overall Delivery Process* requires data items to specify:

- The designation of the activity tree.
- The root activity of the tree for delivery. An XPath expression specifying the root node as a part of a larger
tree. Specifying a root permits delivery to be applied to part of a larger tree structure. The root for delivery
may be the same as the root of the activity tree.
- The activity for delivery. An XPath expression specifying a node within the activity tree.
- The sequencing thread. An integer.
- The learner.

**Process:**

The *Overall Delivery Process* is defined above (section DB.1.3).

**Results:**

The *Overall Delivery Process* returns the following upon completion:

- A Boolean indication if the *Overall Delivery Process* was successful
- An indication of the content resource to deliver if the process was successful
  - The content if the *Overall Delivery Process* was successful, identified by the resource GUID from
    the content package.
  - Error content if the delivery process was not successful.
- An error indication if the process was not successful. Possible errors include:
  - Data errors
    - The activity tree was not properly specified
    - The root of the tree was not properly specified
    - The end of the delivery path was not properly specified
    - The thread was not properly specified (e.g., not 1, or there was no path from the root to
      the end activity for the specified thread)
    - The learner was not properly specified
  - Processing Errors
    - Error in the *Sequencing Rules Process*
    - Error in the *Limit Conditions Process*
    - Error in the *Overall Delivery Process*
The process description does not currently define how to return multiple error indicators. The results description does not indicate how error content or error indicators are defined or bound to a data type.

**Related Services:**
The Overall Delivery Process relies on the following other services:
- Time service (return current time point).

**DB.7.2 Content Delivery Environment Process SAP**

**Request:**
The Content Delivery Environment Process requires data items to specify:
- The designation of the activity tree.
- The root activity of the tree for delivery. An Xpath expression specifying the root node as a part of a larger tree. Specifying a root permits delivery to be applied to part of a larger tree structure. The root for delivery may be the same as the root of the activity tree.
- The activity for delivery. An Xpath expression specifying a node within the activity tree.
- The sequencing thread. An integer.
- The learner.

**Process:**
The Content Delivery Environment Process is defined above (section DB.2).

**Results:**
The Content Delivery Environment Process returns the following upon completion:
- A Boolean indication if the Content Delivery Environment Process was successful.
- An error indication if the process was not successful. Possible errors include:
  - Data errors:
    - The activity tree was not properly specified.
    - The root of the tree was not properly specified.
    - The end of the delivery path was not properly specified.
    - The thread was not properly specified (e.g., not 1, or there was no path from the root to the end activity for the specified thread).
    - The learner was not properly specified.
  - Processing Error:
    - Error in the Content Delivery Environment Process

The process description does not currently define how to return multiple error indicators.

**Related Services:**
The delivery process relies on the following other services:
- Time service (return current time point).
- Tracking Model Services (used to access and set values in the Tracking Model).
RB. Rollup Behavior Model (Just Rollup V0.9)

The SS process uses information about the results of a learner’s interactions with activities, and the learner’s record for objectives (e.g., completion, score) to control the sequencing of other activities. The data attributes that describe the results of the learner’s interactions, i.e., the elements of the tracking model, may be referenced for any activity in the activity tree. The results data for an activity may be determined from the results data from the children of the activity. The process of computing the results data for an activity from the results data from the children of the activity is called the “rollup process” or just “rollup”.

The rollup process is used to maintain the tracking model data for all activities and objectives in the activity tree. The overall sequencing process (see Overall Sequencing Process Documents OP) invokes the rollup process to update the tracking model data when an activity attempt terminates. Other processes may invoke the rollup process at other times, but such invocation is not detailed in this specification.

The overall sequencing process (see Sequencing Process Documents SP) relates the rollup process to the navigation, sequencing, exit, and delivery processes.

The rollup process is applied to all parts of the tracking model:
- **Objective Information** – information about the results of the learner’s interactions related to an objective.
- **Progress Information** – information about a learner’s attempt at an activity.

The rollup process is controlled by parts of the sequencing definition model:
- **Rollup Controls** – controls which data from an activity contributes to the rollup of the parent activity.
- **Rollup Rule Definitions** – rules that indicate how tracking data from an activity or its associated objectives is produced from the results of the children of the activity.
- **Objective Description** – the learning objective associated with an activity.
- **Delivery Controls** – controls when progress and objective data for an activity is recorded.

The behavior of the rollup process is defined in terms of different processes, each of which is applied to one part of the tracking model:
- **Measure Rollup Process** – determines the normalized measure (e.g., score) of the objective associated with an activity.
- **Objective Rollup Process** – determines the objective status of the objective associated with an activity.
- **Activity Progress Rollup Process** – determines the completion status of an activity.
- **Duration Rollup Process** – determines the time spent on an activity.

These individual processes are part of the overall process that propagates results up the activity tree. Each individual process is applied only if enabled by the corresponding Rollup Control elements in the Sequencing Definition Model. Rollup of both Objective and Progress values is applied only to Tracked activities, but Duration Rollup is applied to all activities. Rollup is not otherwise conditional on any other sequencing rules or conditions.

RB.1 Rollup Behavior

The rollup behavior describes how a sequencing system interprets the elements of the sequencing definition model in combination with instance data from the tracking model to update the tracking model.

An implementation must be capable of representing the processes described and have the implemented process exhibit the behavior described. There are no additional requirements on implementing the rollup behavior model.
The rollup behavior relies on the data descriptions from the sequencing definition model (see Sequencing Definition Model Document SM) and the tracking model (see Tracking Model Document TM). These data models also specify default data values or data inheritance in the activity tree that govern the access to activity or tracking data.

**RB.1.1 Measure Rollup Process**

Measure rollup is used to determine the normalized measure of the objective associated with an activity from the values of the measures from the objectives associated with the children of the activity.

Measure Rollup computations (i.e., computing the values of Objective Measure Status and Objective Normalized Measure in the Tracking Model) are controlled by the Rollup Controls value of Rollup Objective Measure Weight and the Delivery Control value of Tracked.

The measure rollup behavior is defined by a weighted measure-based computation.

1. All Tracked children with a positive Rollup Objective Measure Weight are included in the computation. If the Objective Measure Status for the objective associated with any child activity is False (i.e., there is no measure), the Objective Measure Status for the objective associated with the activity is False. Otherwise, compute a weighted average measure across all children of the objective associated with the activity. The computation combines the Objective Normalized Measure for the objective associated with the child activity weighted by the Rollup Objective Measure Weight for the child activity.

The sum of the Rollup Objective Measure Weight values for the child activities need not total to 1.0, but may not exceed 1.0.

The Measure Rollup Process for an activity is specified in the following pseudo code. The pseudo code describes only the measure rollup logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The Measure Rollup Process only describes the expected behavior that an implementation will exhibit.

**Measure Rollup Behavior.**
1. Initialize Total Weighted Measure as zero
2. Initialize Counted Measures as zero
3. For each child of the activity
   3.1. If the Delivery Control attribute value Tracked for the child activity is True Then
   3.1.1. If the Rollup Control attribute value Rollup Objective Measure Weight for the child activity is zero Then
   3.1.1.1. Set the Objective attribute value Objective Measure Status for the objective associated with the activity to False
   3.1.2. Exit Measure Rollup
   3.1.3. If the Rollup Control attribute value Rollup Objective Measure Weight for the activity is not zero Then
   3.1.3.1. Increment Counted Measures by one
   3.1.3.2. Add the product of Objective attribute value Objective Normalize Measure for the objective associated with the child activity multiplied by the Rollup Control attribute value Rollup Objective Measure Weight for the child activity to the Total Weighted Measure
4. If Counted Measures is zero Then
   4.1. Set the Objective attribute value Objective Measure Status for the objective associated with the activity to False
   4.2. Exit Measure Rollup Process
5. If Counted Measures is positive Then
   5.1. Set the Objective attribute value Objective Measure Status for the objective associated with the activity to True
   5.2. Set the Objective attribute value Objective Normalized Measure for the objective associated with the activity to Total Weighted Measure divided by Counted Measure
5.3 **Exit Measure Rollup Process**

### RB.1.2 Objective Rollup Process

Objective rollup is used to determine the objective status of the objective associated with an activity from the values of the objective data for the objectives associated with the children of the activity.

Objective Rollup computations (i.e., computing the values of *Objective Data Status* and *Objective Satisfied Status* in the Tracking Model) are controlled by the Rollup Controls value of *Rollup Objective Satisfied Status* and the Rollup Rules for the activity.

The objective rollup behavior is defined by a measure-based computation, a rule-based computation or an alternative default rule-based computation, applied in that order.

1. **Measure-Based Computation** – The objective is satisfied (set values for *Objective Data Status* and *Objective Satisfied Status*) if the *Objective Normalized Measure* for the objective associated with the activity is equal to or greater than the *Objective Minimum Satisfied Normalized Measure* for the activity. Measure-based computation is used only if the *Objective Satisfied by Measure* attribute for the activity is True.

2. **Rule-Based Computation** – Rollup Rules with Rollup Actions of Failed or Passed are evaluated for the activity. Child activities that are part of the child set (i.e., child activities that have a Rollup control Rollup Objective Satisfied value of True) are used to evaluate the Rollup Condition of the rule. No rules are evaluated if there are no child activities that have a Rollup Objective Satisfied Control value of True. All child activities identified as Tracked are used to evaluate the Rollup Rule. If the Rollup Rule evaluates to True, the Rollup Action sets values for *Objective Data Status* and *Objective Satisfied Status* for the objective associated with the activity. All rules with Rollup Action of Not Satisfied are evaluated before evaluating any rules with Rollup Action of Satisfied. If no rules fire, there is no change to the values of *Objective Data Status* and *Objective Passed Status* for the objective associated with the activity.

3. **Default Rule-Based Computation** – The default rule has a Rollup Child Set value of All, a Rollup Condition value of Satisfied, and a Rollup Action value of Satisfied. The rule-based computation described above is applied using this rule.

The *Objective Rollup Process* for an activity is specified in the following pseudo code. The pseudo code describes only the objective rollup logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The *Objective Rollup Process* only describes the expected behavior that an implementation will exhibit.

**Objective Rollup Behavior Using Measure.**

1. **If** the Objective attribute value *Objective Satisfied by Measure* for the activity is **True Then**
   1.1. **If** the Objective attribute value *Objective Measure Status* for the objective associated with the activity is **False Then**
     1.1.1. Set the Objective attribute value *Objective Data Status* for the objective associated with the activity to False
     1.1.2. **Exit Objective Rollup Process**
   1.2. **If** the Objective attribute value *Objective Measure Status* for the objective associated with the activity is **True Then**
     1.2.1. **If** the Objective attribute value *Objective Normalized Measure* for the objective associated with the activity is equal to or greater than the Objective Description attribute value *Objective Minimum Satisfied Normalized Measure* for the activity **Then**
       1.2.1.1. Set the Objective attribute value *Objective Data Status* for the objective associated with the activity to True
       1.2.1.2. Set the Objective attribute value *Objective Satisfied Status* for the objective associated with the activity to True
       1.2.1.3. **Exit Objective Rollup Process**
1.2.2. If the Objective attribute value Objective Normalized Measure for the objective associated with the activity is less than the Objective Description attribute value Objective Minimum Satisfied Normalized Measure for the activity Then

1.2.2.1. Set the Objective attribute value Objective Data Status for the objective associated with the activity to True

1.2.2.2. Set the Objective attribute value Objective Satisfied Status for the objective associated the activity to False

1.2.2.3. Exit Objective Rollup Process

Objective Rollup Behavior Using Rules.

2. If there are Rollup Rules for the activity where the attribute value Rollup Action is Satisfied or Not Satisfied Then

2.1. For each Rollup Rule for the activity where the attribute value Rollup Action is Not Satisfied

2.1.1. Initialize Activity Children Status Bag as an empty collection

2.1.2. For each child of the activity

2.1.2.1. If the Delivery Control attribute value Tracked for the child activity is True Then

2.1.2.1.1. If the Rollup Control attribute value Rollup Objective Satisfied for the child activity is True Then

2.1.2.1.1.1. Evaluate the Rollup Condition for the rule by applying the Rollup Rule attribute value Rollup Condition for the activity to the appropriate corresponding data associated with the objective of the child activity

2.1.2.1.1.2. If the Rollup Condition for the rule evaluates to True Then

2.1.2.1.1.2.1. Add a True value to the Activity Children Status Bag

2.1.2.1.1.3. If the Rollup Condition for the rule evaluates to False Then

2.1.2.1.1.3.1. Add a False value to the Activity Children Status Bag

2.1.3. If the Activity Children Status Bag is empty Then

2.1.3.1. Exit Objective Rollup Process

2.1.4. Apply the condition from the Rollup Control attribute value Rollup Child Set for the activity to the Activity Children Status Bag to produce a single combined result for the children of the activity

2.1.5. If the combination condition applied to the Rollup Child Set evaluates to True Then

2.1.5.1. Set the Objective attribute value Objective Data Status for the objective associated with the activity to True

2.1.5.2. Set the Objective attribute value Objective Satisfied Status for the objective associated the activity to False

2.2. For each Rollup Rule for the activity where the Rollup Action is Satisfied

2.2.1. Initialize Activity Children Status Bag as an empty collection

2.2.2. For each child of the activity

2.2.2.1. If the Delivery Control attribute value Tracked for the child activity is True Then

2.2.2.1.1. If the Rollup Control attribute value Rollup Objective Satisfied for the child activity is True Then

2.2.2.1.1.1. Evaluate the Rollup Condition by applying the Rollup Rule attribute value Rollup Condition for the activity to the appropriate corresponding data associated with the objective of the child activity

2.2.2.1.1.2. If the Rollup Condition for the rule evaluates to True Then

2.2.2.1.1.2.1. Add a True value to the Activity Children Status Bag

2.2.2.1.1.3. If the Rollup Condition for the rule evaluates to False Then

2.2.2.1.1.3.1. Add a False value to the Activity Children Status Bag

2.2.3. If the Activity Children Status Bag is empty Then

2.2.3.1. Exit Objective Rollup Process

2.2.4. Apply the condition from the Rollup Control attribute value Rollup Child Set for the activity to the Activity Children Status Bag to produce a single combined result for the children of the activity

2.2.5. If the combination condition applied to the Rollup Child Set evaluates to True Then

2.2.5.1. Set the Objective attribute value Objective Data Status for the objective associated with the activity to True
2.2.5.2. Set the Objective attribute value *Objective Satisfied Status* for the objective associated the activity to *True*

2.3. Exit Objective Rollup Process

**Objective Rollup Behavior Using a Default Rule.**

3. If there are NO Rollup Rules for the activity where the attribute value *Rollup Action* is *Satisfied* or *Not Satisfied* Then

3.1. Initialize *Activity Children Status Bag* as an empty collection

3.2. For each child of the activity

3.2.1. If the Delivery Control attribute value *Tracked* for the child activity is *True* Then

3.2.1.1. If the Rollup Control attribute value *Rollup Objective Satisfied Status* for the child activity is *True* Then

3.2.1.1.1. Evaluate the Rollup Condition for the rule by applying a *Satisfied* Condition to the appropriate corresponding data associated with the objective of the child activity

3.2.1.1.2. If the Rollup Condition for the rule evaluates to *True* Then

3.2.1.1.2.1. Add a *True* value to the *Activity Children Status Bag*

3.2.1.2. If the Rollup Condition for the rule evaluates to *False* Then

3.2.1.2.1.1. Add a *False* value to the *Activity Children Status Bag*

3.3. If the *Activity Children Status Bag* is empty Then

3.3.1. Exit Objective Rollup Process

3.4. Apply an *All* combinations to the *Activity Children Status Bag* to produce a single combined result for the children of the activity

3.5. If the combination condition applied to the *Rollup Child Set* evaluates to *True* Then

3.5.1. Set the Objective attribute value *Objective Data Status* for the objective associated with the activity to *True*

3.5.2. Set the Objective attribute value *Objective Satisfied Status* for the objective associated the activity to *True*

**RB.1.3 Activity Progress Rollup Process**

Activity Progress rollup is used to determine the progress and completion status of an activity from the values of the children of the activity.

Activity Progress computations (i.e., computing the values of *Activity Completion Status* and *Activity Completion Amount* in the Tracking Model) are controlled by the Rollup Controls value of *Rollup Progress Completion* and the Rollup Rules for the activity.

The activity progress rollup behavior is defined by a rule-based computation or an alternative default numeric computation, applied in that order.

1. **Rule-Based Computation** – Rollup Rules with *Rollup Actions of Incomplete* or *Completed* are evaluated for the activity. Child activities that are part of the child set (i.e., child activities that have a Rollup control *Rollup Activity Completion* value of *True*) are used to evaluate the *Rollup Condition* of the rule. No rules are evaluated if there are no child activities that have a *Rollup Activity Completion* Control value of *True*. All identified *Tracked* child activities are used to evaluate the Rollup Rule. If the Rollup Rule evaluates to *True*, the *Rollup Action* sets values for *Activity Completion Status* and *Activity Completion Amount* for the activity. All rules with *Rollup Action of Incomplete* are evaluated before evaluating any rules with *Rollup Action of Completed*. If no rules fire, there is no change to the values of *Activity Completion Status* and *Activity Completion Amount* for the activity.

2. **Default Numeric Computation** – If there are no Rollup Rules with *Rollup Actions of Completed* or *Incomplete*, the *Activity Completion Status* is set to *True* and *Activity Completion Amount* is computed as the average of the *Activity Completion Amount* of all *Tracked* child activities that are part of the child set. This computation only applies if all of the child activities in the child set have an *Activity Completion Status* value of *True*, i.e., the
parent activity is assigned a value for completion only if all the children have been completed. If any children have an Activity Completion Status value of False, there is no change to the values of Activity Completion Status and Activity Completion Amount for the activity.

Note, the computational alternative is applied only if the corresponding rollup rules do not exist, independent of the rules firing.

The Activity Progress Rollup Process for an activity is specified in the following pseudo code. The pseudo code describes only the activity progress rollup logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The Activity Progress Rollup Process only describes the expected behavior that an implementation will exhibit.

Activity Progress Rollup Behavior Using Rules.
1. If there are Rollup Rules for the activity where the attribute value Rollup Action is Completed or Incomplete
   Then
   1.1. For each Rollup Rule for the Activity where the attribute value Rollup Action is Incomplete
      1.1.1. Initialize Activity Children Status Bag as an empty collection
      1.1.2. For each child of the activity
      1.1.2.1. If the Delivery Control attribute value Tracked for the child activity is True Then
      1.1.2.1.1. If the Rollup Control attribute value Rollup Activity Completion for the child activity is True Then
      1.1.2.1.1.1. Evaluate the Rollup Condition for the rule by applying the Rollup Rule attribute value Rollup Condition for the activity to the appropriate corresponding data associated with the child activity
      1.1.2.1.1.2. If the Rollup Condition for the rule evaluates to True Then
      1.1.2.1.1.2.1. Add a True value to the Activity Children Status Bag
      1.1.2.1.1.3. If the Rollup Condition for the rule evaluates to False Then
      1.1.2.1.1.3.1. Add a False value to the Activity Children Status Bag
      1.1.3. If the Activity Children Status Bag is empty Then
      1.1.3.1. Exit Activity Rollup Process
      1.1.4. Apply the condition from the Rollup Control attribute value Rollup Child Set for the activity to the Activity Children Status Bag to produce a single combined result for the children of the activity
      1.1.5. If the combination condition applied to the Rollup Child Set evaluates to True Then
      1.1.5.1. Set the Activity Progress attribute value Activity Completion Status for the activity to True
      1.1.5.2. Set the Activity Progress attribute value Activity Completion Amount for the activity to a value less than 0.99999
   1.2. For each Rollup Rule for the activity where the Rollup Action is Completed
   1.2.1. Initialize Activity Children Status Bag as an empty collection
   1.2.2. For each child of the activity
   1.2.2.1. If the Delivery Control attribute value Tracked for the child activity is True Then
   1.2.2.1.1. If the Rollup Control attribute value Rollup Activity Completion for the child activity is True Then
   1.2.2.1.1.1. Evaluate the Rollup Condition by applying the Rollup Rule attribute value Rollup Condition for the activity to the appropriate corresponding data associated with the child activity
   1.2.2.1.1.2. If the Rollup Condition for the rule evaluates to True Then
   1.2.2.1.1.2.1. Add a True value to the Activity Children Status Bag
   1.2.2.1.1.3. If the Rollup Condition for the rule evaluates to False Then
   1.2.2.1.1.3.1. Add a False value to the Activity Children Status Bag
   1.2.3. If the Activity Children Status Bag is empty Then
   1.2.3.1. Exit Activity Rollup Process
   1.2.4. Apply the condition from the Rollup Control attribute value Rollup Child Set for the activity to the Activity Children Status Bag to produce a single combined result for the children of the activity
   1.2.5. If the combination condition applied to the Rollup Child Set evaluates to True Then
1.2.5.1. Set the Activity Progress attribute value *Activity Completion Status* for the activity to True

1.2.5.2. Set the Activity Progress attribute value *Activity Completion Amount* for the activity to 1.0

1.3 Exit Activity Rollup Process

**Activity Progress Rollup Behavior Using Completion Data.**

2. **If** there are NO Rollup Rules for the activity where the *Rollup Action* attribute is *Completed* or *Incomplete* **Then**

2.1. Initialize *Activity Children Status Bag* as an empty collection

2.2. Initialize *Activity Children Value Bag* as an empty collection

2.3. **For** each child of the activity

2.3.1. **If** the Delivery Control attribute value *Tracked* for the child activity is *True* **Then**

2.3.1.1. **If** the Rollup Control attribute value *Rollup Activity Completion* for the child activity is *True* **Then**

2.3.1.1.1. Add the Tracking Model attribute value *Activity Completion Status* for the child activity to the *Activity Children Status Bag*

2.3.1.1.2. Add the Tracking Model attribute value *Activity Completion Amount* for the child activity to the *Activity Children Value Bag*

2.4. **If** the *Activity Children Status Bag* is empty **Then**

2.4.1. Exit Activity Rollup Process

2.5. **If** the *Activity Children Status Bag* contains any *False* values **Then**

2.5.1. **Set** the Activity Progress attribute value *Activity Completion Status* for the activity to *False*

2.5.2. **Exit** Activity Rollup Process

2.5.3. **If** the *Activity Children Status Bag* contains only *True* values **Then**

2.5.1. **Set** the Activity Progress attribute value *Activity Completion Status* for the activity to *True*

2.5.2. **Set** the Activity Progress attribute value *Activity Completion Amount* for the activity to be the average of the values in the *Activity Value Bag*

2.5.3. **Exit** Activity Rollup Process

**RB.1.4 Duration Rollup Process**

Duration rollup is used to determine the activity duration of an activity from the values of the durations of the children of the activity.

Duration Rollup computations (i.e., computing the values of *Activity WallClock Duration* and *Activity Interaction Duration* in the Tracking Model) are controlled by the Rollup Control attribute value of *Rollup Progress Duration*.

The duration rollup behavior is defined by a numeric computation.

1. **Duration Computation** – The activity duration (values for *Activity WallClock Duration* and *Activity Interaction Duration*) are summed from the activity durations of the children of the activity. Activities are included in the sum only if the Rollup Control attribute value of *Rollup Progress Duration* for the child is *True*.

Duration rollup is unconditionally performed for all activities.

The *Duration Rollup Process* for an activity is specified by the following pseudo code. The pseudo code describes only the duration rollup logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The *Duration Rollup Process* only describes the expected behavior that an implementation will exhibit.

**Duration Rollup Behavior.**

1. **Initialize Wallclock Duration Sum** value to zero

2. **Initialize Interaction Duration Sum** value to zero

3. **Initialize Counted Children** as zero

© Copyright 2002, Carnegie Mellon University, All Rights Reserved. 7
4. **For** each child of the activity
   4.1. **If** the Rollup Control attribute value **Rollup Progress Duration** for the child activity is **True** **Then**
       4.1.1. Increment **Counted Children** by one
       4.1.2. Add the Activity Progress attribute value **Activity WallClock Duration** for the child activity to **Wallclock Duration Sum**
       4.1.3. Add the Activity Progress attribute value **Activity Interaction Duration** for the child activity to **Interaction Duration Sum**

5. **If** **Counted Children** is positive **Then**
   5.1. Set the Activity Progress attribute value **Activity WallClock Duration** for the activity to **Wallclock Duration Sum**
   5.2. Set the Activity Progress attribute value **Activity Interaction Duration** for the activity to **Interaction Duration Sum**

There is no defined behavior to rollup attempt data. Attempt duration values (**Attempt WallClock Duration** and **Attempt Interaction Duration**) are maintained for all activities in the activity tree. Activity attempt duration values include the durations of all children of the activity.

**RB.1.5 Overall Rollup Process**

The **Overall Rollup Process** is used to propagate all results upward through the entire activity tree. The rollup process is applied to:

- An activity tree, designated by the root activity of the tree.
- The activity within the tree that terminated that triggered rollup, i.e., the activity whose tracking data has changed.
- The sequencing thread.
- An indication of the learner.

The root of the tree and the activity within the tree specify a unique single path through the tree, denoted the rollup path. Rollup is applied to all activities along that path, starting from the parent of the activity that triggered the rollup and proceeding backward along the path of parents of the activities to the root of the activity tree.

At each step along the rollup path, all rollup processes, **Measure Rollup**, **Objective Rollup**, **Activity Progress Rollup** and **Duration Rollup**, are applied. **Measure Rollup** must be applied before any other rollup process, but there are no other ordering constraints on the rollup processes. Whenever the rollup process is initiated, it is performed on all activities along the path; the process does not stop if rollup does not change any values for an activity (other actions may have changed values of other children along different paths that contribute to the rolled up values at some level in the tree).

The **Overall Rollup Process** is defined only for the main sequencing thread (**Thread ID** 1).

The **Overall Rollup Process** is specified by the following pseudo code. The pseudo code describes only the rollup logic. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The **Overall Rollup Process** only describes the expected behavior that an implementation will exhibit.

**Overall Rollup Behavior**

1. Form the **Activity Path** as an ordered sequence of all activities from the root of the activity tree to the activity that triggered rollup
2. Eliminate the activity that triggered rollup from the **Activity Path**
3. **If** any activity on the **Activity Path** does not have a **Thread ID** of one **Then**
   3.1. **Exit** **Overall Rollup Process**
4. **If** the **Activity Path** is empty **Then**
   4.1. **Exit** **Overall Rollup Process**
5. **For** each activity in the **Activity Path**, in reverse order
5.1. Apply the Measure Rollup Process to the activity
5.2. Apply the Objective Rollup Process to the activity
5.3. Apply the Activity Progress Rollup Process to the activity
5.4. Apply the Duration Rollup Process to the activity

RB.2 Notes/Changes from IMS SS Public Draft

This model is identical in functionality and intent with that described in the IMS SS Public Draft. Descriptions and terminology have been simplified. Inconsistencies and partially specified behaviors have been fully specified.

References to SS data, application profile–specific behaviors and other descriptions not pertaining to just the rollup behavior have been eliminated. This part just describes the rollup behavior itself, not how rollup is used by the sequencing process.

The following behaviors have been changed.

Measure Rollup:
- The procedure is defined in terms of using only the measure weight to control what elements are included in the rollup. Measure value requirement is not considered, as it is not reliable.

Objective Rollup:
- The condition of skipping the rule evaluation if the child set is empty was added.

Activity Progress Rollup:
- The condition of skipping the rule evaluation if the child set is empty was added.

Duration Rollup:
- The rollup behavior for in progress activities has been dropped. How to properly describe the tracking of in progress activities is not clear.

RB.3 Open Issues

Measure rollup uses a zero weight to exclude items from the rollup. It may be desirable to include items, but not to weight them. While this can be done with a small error by setting a very small rollup value (0.00001), the proper solution is to add a Boolean rollup control that explicitly indicates if a child is to participate in rollup. [SS.RB.01]

The measure-based objective rollup is to be applied if the Objective Passed by Measure attribute is True. This attribute is associated with an activity, not with the objective associated with the activity. This implies that different activities can define different passing measures for the same objective. Providing consistent scoring requires each activity to have a measure. Associating the passing measure with the objective would provide consistency, but each activity would then need an explicit control attribute to use activity rules to override the measure-based rollup process. [SS.RB.02]

The default objective rollup rule could be eliminated by changing the defaults in the data model to have a single passed rule instead of a default rule that always fails. [SS.RB.03]

The Delivery Control Tracked is applied to the children on the activity. The control is used when rolling up the children in the objective rollup. When the objective rollup is computed directly from the measure, Tracked does not control the process. Should measure-based objective rollup always be performed for activities that are not being tracked? [SS.RB.04]

Direct computation of objective results (see RB.1.3) is the only part of the rollup process that is not an actual rollup of child values. This behavior should more properly be an activity exit behavior rather than the direct computation, which confuses how exit propagates up the activity tree. [SS.RB.05]
Activity durations are summed assuming the children are not active. Is there a case when the rollup is triggered (e.g., suspend) and the children are still active? If so, the time must be computed as the difference between the current time and the start time. The tracking model does not contain the necessary data for such a computation. Similarly, the rollup process implies that attempt data includes start times for attempts, but this data is also not a part of the tracking model. An alternative set of tracking data may be required to determine the activity durations for in progress attempts. The full tracking model that would be needed to compute durations is not obvious. [SS.RB.06]

**RB.4 Implementation Recommendations**

Values of Completion and Measure are normalized between 0..1. Testing for the exact value of 1.0 is unreliable. Any value > 0.99999 should be considered equal to 1.0 in evaluating rule conditions (based on the required precision of 5 decimal digits).

Normalized data values are described as value space real. A decimal data type should be considered as an alternative to a float data type for the implementation to maintain the required precision.

Rule processors should use short circuit evaluation techniques when determining if the rule conditions are true. For example, assume an And rule contains a condition on both Objective Data Status and Objective Satisfied Status. If the Objective Data Status evaluates to False, the condition on the the value of Objective Satisfied Status should not be evaluated (in this case, the Objective Satisfied Status value is unreliable). The rule processor may reorder conditions to enable short circuit evaluation. However, short circuit evaluation is not required, as the default values insure that the same sequencing behaviors will result whether short circuit evaluation is not used or not.

Activity Completion Amount has a defined precision. When the value for an activity is computed from the value of the children, the required precision must be maintained. It is permissible to round the result after the computation to maintain just the required minimum precision.

The implementation must ensure that data is updated in the proper sequence. For example, the value of Objective Data Status should not be set to True until there is a value of Objective Satisfied Status. Reversing the order of assignment of these two values could result in a situation where an accessor uses a True value of Objective Data Status to determine if it can access Objective Satisfied Status, and then accesses Objective Satisfied Status before the second assignment is complete. Similarly, the implementation must ensure that data values are consistent when being accessed, i.e., the values of the set of activities that participate in a child set must be consistent while the entire set of rules applied during rollup are evaluated. Implementing proper transactional controls are left to the implementation.

Processes use data values from the sequencing definition model. Each attribute of the model has a value or inherits the value of the parent activity as specified in the sequencing definition model. In addition, there is a defined default value for each attribute. Processes that access data must access the proper value, i.e., the value of the attribute for the activity or the value for the parent if the data does not exist and is inherited. The implementation may fully instantiate the activity tree with values or provide an inheritance mechanism, either in the data model itself or as part of the rollup process. The choice of approach is left to the implementation, which must only provide the proper data for use in the rollup process.

**RB.5 Bindings**

Not applicable to a behavior model.

**RB.6 Rollup Process Service**

The rollup process may be implemented as a service. The rollup process has one service access point (SAP) for the Overall Rollup Process.
RB.6.1 Overall Rollup Process SAP

Request:
The Overall Rollup Process requires data items to specify:
- The designation of the activity tree.
- The root activity of the tree for rollup. An XPath expression specifying the root node as a part of a larger tree. Specifying a root permits rollup to be applied to part of a larger tree structure. The root for rollup may be the same as the root of the activity tree.
- The activity at the end of the rollup path. An XPath expression specifying a node within the activity tree.
- The sequencing thread. An integer.
- The learner.

Process:
The Overall Rollup Process is defined above (section RB.1.5).

Results:
The Overall Rollup Process returns the following upon completion:
- A Boolean indication if the Overall Rollup Process was successful.
- An error indication if the process was not successful. Possible errors include:
  - Data errors:
    - The activity tree was not properly specified.
    - The root of the tree was not properly specified.
    - The end of the rollup path was not properly specified.
    - The thread was not properly specified (e.g., not 1, or there was no path from the root to the end activity for the specified thread).
    - The learner was not properly specified.
  - Processing Errors:
    - Error in the Measure Rollup Process.
    - Error in the Objective Rollup Process.
    - Error in the Activity Progress Rollup Process.
    - Error in the Duration Rollup Process.
    - Error in the Overall Rollup Process.

The results description does not indicate how error indicators are defined or bound to a data type.

Related Services:
The Overall Rollup Process relies on the following other services:
- Tracking Model Services (used to access and set values in the Tracking Model).
OP. Overall Sequencing Process (Just Workflow V0.5)

The overall sequencing process combines all of the other elements of SS to produce the complete sequencing, delivery and control of learning experience and relates all of other SS processes and behaviors.

- **Navigation Behavior Model** – The SS process that evaluates a navigation request and determines the exit and sequencing requests that should be processed to identify and deliver content to the learner (see Navigation Behavior Document NB).
- **Exit Behavior Model** – The SS process that processes an exit request to terminate or suspend an activity or set of activities (see Exit Behavior Document EB).
- **Sequencing Behavior Model** – The SS process that evaluates a sequencing request in terms of the content model described by the activity tree and determines the actual content object that should be delivered to the learner (see Sequencing Behavior Document SB).
- **Delivery Behavior Model** – The SS process that validates that the content resources for the identified activity may be delivered, i.e., all of the conditions that apply to the delivery of the content for the activity and attempt still hold (see Delivery Behavior Document DB).
- **Rollup Behavior Model** – The SS process that computes the results data for an activity from the results data from the children of the activity (see Rollup Behavior Document RB).

The overall sequencing process describes only a logical workflow. It makes no assumption on how or when the various steps of the process are invoked, or in what order they are processed.

The behavior of the sequencing process is defined in terms of different processes:

- **Navigation Request Process** – processes the navigation request to identify the corresponding sequencing and exit requests.
- **Exit Request Process** – processes the exit request to set the corresponding state and activity data.
- **Sequence Request Process** – processes the sequencing request that controls all sequencing behavior, combining:
  - **Start Sequencing Request Process** – processes a Start sequencing request.
  - **Continue Sequencing Request Process** – processes a Continue sequencing request.
  - **Continue From Parent Sequencing Request Process** – processes a Continue From Parent sequencing request.
  - **Previous Sequencing Request Process** – processes a Previous sequencing request.
  - **Choice Sequencing Request Process** – processes a Choice sequencing request.
  - **Retry Sequencing Request Process** – processes a Retry sequencing request.
  - **Retry All Sequencing Request Process** – processes a Retry All sequencing request.
  - **Exit Sequencing Request Process** – processes an Exit sequencing request.
  - **Exit Parent Sequencing Request Process** – processes an Exit Parent sequencing request.
  - **Exit All Sequencing Request Process** – processes an Exit All sequencing request.
- **Overall Delivery Process** – the overall process that controls all delivery behavior, combining:
  - **Sequencing Rules Process** – applies certain precondition sequencing rules to determine if the delivery request is valid.
  - **Limit Conditions Process** – applies limit conditions to determine if the delivery request is valid.
- **Overall Rollup Process** – the overall process that propagates results up the activity tree, combining:
  - **Measure Rollup Process** – determines the normalized measure (e.g., score) of the objective associated with an activity.
  - **Objective Rollup Process** – determines the objective status of the objective associated with an activity.
  - **Activity Progress Rollup Process** – determines the completion status of an activity.
  - **Duration Rollup Process** – determines the time spent on an activity.
- **Content Delivery Environment Process** – the process that manages the actual delivery of content resources to the learner.
OP.1 Overall Behavior

The overall behavior functions within the scope of some Learning Technology System (LTS), e.g., an LMS, that has identified learners and internal representations of content activity trees, sequencing descriptions, content resources and data models used to track the learner and record state. How these actual data models are implemented or initialized is not specified.

An implementation must be capable of representing the processes described and have the implemented processes exhibit the behavior described. There are no additional requirements on implementing the overall sequencing behavior.

When the learner first interacts with the learning experience, the LTS will generate a start navigation request. Other interactions with the LTS will result in other navigation requests. How or when a navigation request is generated is not detailed.

The sequencing system operates in a loop, awaiting external navigation requests, processing these, possibly delivering content, and then waiting for additional requests.

The sequencing system continues operations until terminated by the controlling LTS.

The navigation requests are translated by the navigation process into exit and sequencing requests. If the learner is experiencing an activity, the navigation request results in the current activity being terminated by processing the exit request by the exit process – only one activity in the main sequencing thread may be active. Sequencing rules in the exit process may override the sequencing request. The rollup process is performed to propagate results from the activity to parent activities in the aggregation. The sequencing request is processed by the sequencing process to determine the next activity in the sequence, i.e., the next activity to present to the learner. If the sequencing process identifies an activity, the designated activity is passed to the delivery process. The delivery process will verify that the activity should be delivered, i.e., that the limit conditions and sequencing rules applied to the activity permit it to be delivered. The content resources associated with the identified activity are presented to the learner, and the learner’s interactions with the content may be tracked by the LTS (how this tracking is done is not specified in SS). Once the content has been delivered to the learner, the Overall Sequencing Process waits for the next navigation request.

The Overall Sequencing Process is specified by the following pseudo code. The pseudo code describes only the processing. How this process is implemented or how information is encoded, stored, represented or bound is outside the scope of this specification. The Overall Sequencing Process only describes the expected behavior that an implementation will exhibit.

**Overall Sequencing Loop**

1. Wait for a Navigation Request
2. Pass the navigation request to the Navigation Request Process (returns an Exit Request and a Sequencing Request)
   2.1. If the Navigation Request Process returns an error Then
       2.1.1. Handle the navigation request error (behavior not specified)
       2.1.2. Loop – wait for the next Navigation Request
3. If there is an Exit Request Then
   3.1. Pass the Exit Request to the Exit Request Process to be applied to the current activity (may change the Deliver Request)
      3.1.1. If the Exit Request Process returns an error Then
          3.1.1.1. Handle the exit request error (behavior not specified)
          3.1.1.2. Loop – wait for the next Navigation Request
4. Apply the Rollup Process to the activity that was terminated
   4.1. If the Rollup Process returns an error Then
4.1.1. Handle the rollup error (behavior not specified)
4.1.2. Loop – wait for the next Navigation Request

5. If there is a Sequencing Request Then
5.1. Pass the sequencing request to the Sequencing Request Process (returns a Delivery Request)
5.1.1. If the Sequencing Request Process returns an error Then
5.1.1.1. Handle the sequence request error (behavior not specified)
5.1.1.2. Loop – wait for the next Navigation Request
5.1.2. If the Sequencing Request Process returns a request to exit sequencing Then
5.1.2.1. Exit Overall Sequencing Loop – sequencing is terminated
5.1.3. If the Sequencing Request Process does not identify an activity for delivery Then
5.1.3.1. Loop – wait for the next Navigation Request

6. If there is a Delivery Request Then
6.1. Pass the Delivery Request to the Delivery Request Process (returns the content resources for the activity to be delivered)
6.1.1. If the Delivery Request Process returns an error Then
6.1.1.1. Handle the delivery request error (behavior not specified)
6.1.1.2. Loop – wait for the next Navigation Request
6.1.2. If the Delivery Request Process does not identify an activity for delivery Then
6.1.2.1. Loop – wait for the next Navigation Request

7. Deliver the content in the Content Delivery Environment Process
8. Loop – wait for the next Navigation Request

The overall behavior description is described in terms of one request being fully processed before another request is received. There is no constraint on the timing of when requests are received. The description does not detail or otherwise make any assumptions as to how the requests are serialized or how any request may preempt or terminate any in-process request.

**OP.2 Other Elements**

This section outlines additional characteristics of the content model and sequencing behaviors that are part of SS.

**OP.2.1 Activity Tree**

Content in SS is organized into a hierarchical structure. Each activity includes one or more child activities. Each activity has an associated set of sequencing behaviors, defined by the sequencing definition mode (see Sequencing Definition Model Document SB). The sequencing behaviors describe how the activity or how the children of the activity are used to created the desired learning experience.

The sequencing process traverses the sequencing tree, applying the sequencing rules, to determine the activities to deliver to the learner. Content resources from the identified activities are delivered to the learner to create the desired learning experience.

SS makes no requirements on the structure, organization or instantiation of the activity tree. The tree and the associated sequencing definitions may be static or they may be dynamically created. How to create and maintain the activity tree and associated sequencing definition is not specified.

**OP.2.2 Objectives**

Learning objectives are separate from learning activities in SS. Learning objectives represent a set of globally scoped data items, each with a satisfaction status and a satisfaction measure. SS makes no assumption as to how to interpret the objective (e.g., is it a competency, is it a mastery, or is it simply a shared value). Multiple activities may reference the same objective, thus sharing the data values.
The scope of objectives and the resolution of objective IDs are not specified. An objective may be shared within a single activity tree or may be shared globally across multiple instantiations of tracking models. The scope of sharing is not specified and is determined by implementations.

**OP.2.3 Sequencing Threads**

In addition to the main sequenced activity tree, there may be additional (scaffold) content and activities available to the learner. There may be multiple activities at each node in the activity tree. Each of the different sets of activities at each level in the activity tree are designated as being a part of a different sequencing thread. There must be a “main” sequencing thread. Other activities are parts of the other sequencing threads.

Sequencing may be applied to any thread, but the specification only details how to sequence the main thread. How the other threads are made available to the learner for navigation and delivery are not specified. How to sequence activities in the other thread (other than selection via a choice) is not specified.

**OP.2.4 Presentation and Environment**

SS makes no assumption as to how content and controls are rendered or presented to the learner (e.g., style, placement, GUI widgets). How events are triggered in the interface environment, how they are communicated to the LTS, and how the LTS delivers content to the learner and the learner’s environment are not specified.

While an external event may trigger a navigation event and resulting navigation and exit requests, the event may trigger other actions. These behaviors are not specified.

**OP.2.5 Suspend/Resume**

An activity may be suspended and later resumed. Certain navigation requests result in an activity or collection of activities being suspended, i.e., the activity is not active, but it is accumulating wall clock time. Other activities may be delivered while the activity is suspended. The activity may be resumed later without counting as a new activity attempt. Additionally, the suspended activity may be abandoned or exited.

**OP.2.6 Start/Stop**

SS does not specify how to start the overall sequencing process or how to stop the process. Generally, the LTS will recognize some event, e.g., a course login, to start sequencing. Some other event, e.g., a logout, is mapped to the appropriate navigation, exit and sequencing requests, after which the LTS may terminate the overall sequencing process.

**OP.2.7 Data Persistence**

SS does not specify how data (e.g., tracking data) is to be persisted across multiple instantiations of the overall sequencing process for a particular learner and activity tree or learning experience, e.g., across multiple login sessions. It is necessary to persist control, tracking and state data.

**OP.2.8 Content Types**

SS does not limit or specify the types of learning content or content resources that are sequenced and delivered to the learner. In particular, SS does not require content use a particular communications to LTS management scheme such as the CMI API.

**OP.2.9 Active/Passive Content**

SS relies on values within the tracking model to control sequencing. SS does not specify how the tracking model values for an activity or an objective are set or updated. SS differentiates between active and passive content and
supports both active and passive content. Active content is responsible for setting elements of the tracking model directly. For passive content, SS will automatically set certain values in the tracking model. SS makes no assumptions about how content actually behaves, e.g., passive content may set values in the tracking model.

**OP.3 Notes/Changes from IMS SS Public Draft**

This model is identical in functionality and intent with that described in the IMS SS Public Draft. Descriptions and terminology have been simplified. Inconsistencies and partially specified behaviors have been fully specified.

References to SS data, application profile specific behaviors and other descriptions not pertaining to just the sequencing behavior have been eliminated. This part just describes the sequencing behavior itself, not how the sequencing process is used within the overall sequencing framework.

This is a new data behavior model. Overall processing is implicit in the IMS SS Public Draft. This model makes the behavior explicit to tie together all of the other behaviors and processes.

**OP.4 Open Issues**

Implementation recommendations repeat elements from other documents and could be deleted from here. [SS.OP.01]

The set of other elements (section OP.2) may not be complete. [SS.OP.02]

The behavior of sequencing threads is not adequately defined. [SS.OP.03]

The overall behavior is based on the assumption that there is no particular need to explicitly evaluate the sequencing rules when any activity terminates, in particular prior to rollup, i.e., there is no need to explicitly trigger post conditions independently of a sequencing request. The assumption is that the navigation behavior will produce a sequencing request that will be processed after rollup, and that sequencing request will evaluate all of the sequencing rules, including the post condition rules. Alternatively, if the assumption is not true, there could be an additional exit/sequencing process to process the post condition rules. [SS.OP.04] (also SS.EB.07)

**OP.5 Implementation Recommendations**

Values of *Completion* and *Measure* are normalized between 0..1. Testing for the exact value of 1.0 is unreliable. Any value > 0.99999 should be considered equal to 1.0 in evaluating rule conditions.

Normalized data values are described as value space real. A decimal data type should be considered as an alternative to a float data type for the implementation.

Rule processors should use short circuit evaluation techniques when determining if the rule conditions are true. For example, assume an *And* rule contains a condition on both *Objective Data Status* and *Objective Satisfied Status*. If the *Objective Data Status* evaluates to False, the condition on the the value of *Objective Satisfied Status* should not be evaluated (in this case, the *Objective Satisfied Status* value is unreliable). The rule processor may reorder conditions to enable short circuit evaluation. However, short circuit evaluation is not required, as the default values insure that the same navigation behaviors will result whether short circuit evaluation is not used or not.

The implementation must ensure that data values are consistent when being accessed, i.e., the values of the set of activities that participate in a child set must be consistent while the entire set of rules applied during navigation are evaluated. Implementing proper transactional controls are left to the implementation.

Processes use data values from the sequencing definition model. Each attribute of the model has a value or inherits the value of the parent activity as specified in the sequencing definition model. In addition, there is a defined default value for each attribute. Processes that access data must access the proper value, i.e., the value of the attribute for
the activity or the value for the parent if the data does not exist and is inherited. The implementation may fully
instantiate the activity tree with values or provide an inheritance mechanism, either in the data model itself or as part
of the sequencing process. The choice of approach is left to the implementation, which must only provide the proper
data for use in the sequencing process.

There are no constraints on the order in with the limit conditions or precondition sequencing rules are applied.

There is no requirement that the constraint checking processes identify that multiple limits and preconditions are
violated. There is no restriction that the constraint checking processes stop after finding the first constraint that is
violated.

**OP.6 Bindings**

Not applicable to a behavior model.

**OP.7 Overall Sequencing Process Service**

The overall sequencing process is just a workflow. While a monolithic implementation could create a complete
sequencing service, the overall content management and delivery will rely on the other services.

The overall sequencing process has three service access points (SAP):

- **Sequencing Initialization Process** – Initialize the sequencing process.
- **Sequencing Loop Process** – Process one navigation request and deliver one activity.
- **Sequencing Termination Process** – Terminate the sequencing process.

**OP.7.1 Sequencing Initialization SAP**

**Request:**
The **Sequencing Initialization Process** requires data items to specify:

- The designation of the activity tree.
- The root activity of the tree for sequencing. An Xpath expression specifying the root node as a part of a
  larger tree. Specifying a root permits sequencing to be applied to part of a larger tree structure. The root
  for sequencing may be the same as the root of the activity tree.
- The learner.

**Process:**
The **Sequencing Initialization Process** is defined above (section OP.1). The process performs all actions to initialize
sequencing. It leaves the sequencing system ready to accept a navigation request. The process does not include a
navigation request.

**Results:**
The **Sequencing Initialization Process** returns the following upon completion:

- A Boolean indication if the **Sequencing Initialization Process** was successful.
- An error indication if the process was not successful. Possible errors include:
  - Data errors:
    - The activity tree was not properly specified.
    - The root of the tree was not properly specified.
    - The learner was not properly specified.
  - Processing Errors:
    - Error in the **Sequencing Initialization Process**.

The process description does not currently define how to return multiple error indicators.

**Related Services:**
The Sequencing Initialization Process relies on the following other services:

- None.

**OP.7.2 Sequencing Loop Process SAP**

**Request:**
The Sequencing Loop Process requires data items to specify:

- The designation of the activity tree.
- The root activity of the tree for sequencing. An Xpath expression specifying the root node as a part of a larger tree. Specifying a root permits sequencing to be applied to part of a larger tree structure. The root for sequencing may be the same as the root of the activity tree.
- The navigation request (from the Navigation Request vocabulary).
- A navigation location. An Xpath expression specifying a node in the activity tree. Optional (only for Choice requests).
- The sequencing thread. An integer.
- The learner.

**Process:**
The Sequencing Loop Process is defined above (section OP.1). It processes one navigation request and may result in the delivery of content resources for an activity to the learner.

**Results:**
The Sequencing Loop Process returns the following upon completion:

- A Boolean indication if the Sequencing Loop Process was successful.
- An error indication if the process was not successful. Possible errors include:
  - Data errors:
    - The activity tree was not properly specified.
    - The root of the tree was not properly specified.
    - The navigation request was not properly specified.
    - The navigation location was not properly specified.
    - The thread was not properly specified (e.g., not 1, or there was no path from the root to the end activity for the specified thread).
    - The learner was not properly specified.
  - Processing Errors:
    - Error in the Sequencing Loop Process.

The process description does not currently define how to return multiple error indicators.

**Related Services:**
The sequencing initialization process relies on the following other services:

- Navigation Request Service.
- Exit Request Service.
- Overall Rollup Process Service.
- Overall Sequencing Process Service.
- Overall Delivery Process Service.
- Content Delivery Environment Service.

**OP.7.3 Sequencing Termination SAP**

**Request:**
The Sequencing Termination Process requires data items to specify:

- The designation of the activity tree.
• The root activity of the tree for sequencing. An XPath expression specifying the root node as a part of a larger tree. Specifying a root permits sequencing to be applied to part of a larger tree structure. The root for sequencing may be the same as the root of the activity tree.
• The learner.

Process:
The Sequencing Termination Process is defined above (section OP.1). The process performs all actions to terminate sequencing. It begins by identifying the current activity for the learner. It issues an Exit All navigation request to the overall sequencing process. Once the navigation request has been completed, sequencing terminated.

Results:
The Sequencing Termination Process returns the following upon completion:
• A Boolean indication if the Sequencing Termination Process was successful.
• An error indication if the process was not successful. Possible errors include:
  o Data errors:
    ▪ The activity tree was not properly specified.
    ▪ The root of the tree was not properly specified.
    ▪ The learner was not properly specified.
  o Processing Errors:
    ▪ Error in the Sequencing Termination Process.

The process description does not currently define how to return multiple error indicators. The process does not define how to terminate content actively being presented to the learner.

Related Services:
The Sequencing Termination Process relies on the following other services:
• Sequencing Loop Process.
E. Extensibility (Just Extensibility V0.5)

E.1 Extensibility

The SS Specification does not define extensibility mechanisms to represent additional sequencing behaviors. The specification does not explicitly exclude extensions.

Implementers are encouraged to explore different or additional sequencing behaviors by defining additional capabilities and by adding elements to the sequencing definition model and describing their associated behaviors. For exchange, the binding should also be modified to include the extensions. Extensions and behaviors that are outside the scope of describing sequencing, e.g., rendering behaviors or run-time control behaviors should be kept separate from sequencing behaviors.

Extensions must not override or disable existing sequencing behaviors. They should be consistent with all of the assumptions that underlie the SS Specification and function within the existing behavior model.

E.2 Notes/Changes from IMS SS Public Draft

Editorial changes only.

E.3 Open Issues

None.

E.4 Bindings

For exchange of sequencing descriptions, the binding should also be modified to include the extensions. Extensions should be placed within their own name space.
C. Conformance (Just Conformance V0.5)

C.1 Behavior Conformance

An implementation using the Simple Sequencing Specification may map the organization of the manifest of the content package (CP) to the elements in its internal representation of the activity tree. The process that implements the behavior model specified in the Specification operates on the internal representation of the organization of the manifest represented in the activity tree. For a given sequence of timing-independent navigation events, the process that implements the behavior model will produce a sequence of activities, and their associated content resources, to be delivered to the learner.

All strictly conforming implementations of the Specification that process a content package that include Simple Sequencing elements within the organization of the manifest shall produce the same sequence of content resources, given the same initial manifest and the same sequence of timing-independent navigation events.

C.2 Notes/Changes from IMS SS Public Draft

Editorial changes only.

C.3 Open Issues

The conformance statement only applies to processors that rely on the content package and manifest. A system that implements SS without using CP or without using the binding would not be covered by the conformance statement, but the desire is that any implementation of the data and behavior models be conforming, independent of the binding for exchange. [SS.C.01]

The document set would benefit from a set of behavior conformance test cases that contain either a manifest or the data describing the activity tree and sequencing behaviors, sets of one or more navigation requests, initial conditions, and the resulting correct ordered set of content resources for the corresponding set of navigation requests. [SS.C.02]

C.4 Bindings

Conformance to the binding is part of the binding description.
G. SS Glossary (Just Glossary V0.5)

The following terms are used throughout the SS document set to describe parts of the data model or sequencing behavior.

**Activity** – A discrete unit of learning that is sequenced. Activities may have associated resources that represent content that can be delivered to the learner. Activities can be organized in aggregations to form higher-level activities and may be composed of multiple levels of sub-activities. **Alternative:** An instantiation of a node in the activity tree for a learner.

**Activity State Model** – An information model containing state data for learner-related interactions with activities.

**Activity Tree** – The hierarchical collection of content objects (activities) with associated rules and specifications of learning behaviors, conditions and limits. The activity tree describes a complex learning experience. **Alternative:** The representation of the parent-child relationships between activities.

**Aggregation** – A collection of activities within the activity tree. A subtree of the activity tree with a root at any activity.

**Attempt** – A tracked interaction of a learner with an activity. The attempt begins when the activity is delivered and continues until the activity terminates without being suspended (see also resume, suspend). An attempt may span multiple sessions (see also session).

**Content Aggregation** – See Aggregation.

**Content Resource** – See Resource.

**Content Package** – A collection of learning content assets and resources, and supporting data, including a manifest, as defined by the IMS Content Packaging Specification.

**Cluster** – A node in the activity tree and all direct (first level) decedents of the node.

**Delivery Behavior Model** – The process that validates that the content resources for the identified activity may be delivered, i.e., none of the conditions that apply to the delivery of the content for the activity and attempt have been or are being violated.

**Delivery Request** – The identification of an activity whose content resources are to be delivered to the learner.

**Exit Behavior Model** – The process that evaluates an exit request to exit (e.g., complete, terminate, suspend) a content activity.

**Exit Request** – The identification of the action to take to exit an activity or set of activities.

**Interaction Duration** – The cumulative amount of time that a learner spends on an activity. The time is measured from the start of the activity to the end of the activity and excludes any time when the activity is suspended (see also WallClock Duration).

**Learner** – The agent for which interaction data is maintained.

**Navigation Behavior Model** – The process that evaluates a navigation request and determines the sequencing and exit requests that should be processed to identify and deliver content to the learner.
Navigation Event – An event resulting from an external user interaction that triggers a navigation request.

Navigation Request – The identification of a navigation action.

Objective – A (global) data item that may be associated with one or more elements of the activity tree. The tracking model maintains certain data items for an objective (for each learner) including an indication of the learner having satisfied the objective and a measure of satisfaction. While the objective may be associated with a learning objective and the tracking data may represent a pass/fail status and a score, the data item can be used for any Boolean state value with an associated optional measure.

Objective Information – Information about the results of the learner’s interactions related to an objective.

Overall Sequencing Process – The overall sequencing process that relates the navigation, exit, sequencing, delivery, and rollup processes.

Progress Information – Information about a learner’s attempt at an activity.

Resource – Content to be delivered to the learner (e.g., web page, media file, text file, assessment object) as defined by the IMS Content Packaging Specification.

Resume – Restarting an activity that had been previously suspended (see also attempt, suspend).

Rollup Behavior Model – The process that computes the results data for an activity from the results data from the children of the activity.


Sequencing Behavior Model – The process that evaluates a sequencing request in terms of the content model described by the activity tree and determines what actual content object should be delivered to the learner.

Sequencing Definition Model – An information model describing intended sequencing behaviors. Rules, limit conditions and sequencing behaviors defined for activities.

Sequencing Request – The identification of a sequencing action.

Session – A learner’s interaction with an activity not interrupted by suspending the activity (see also suspend).

Suspend – Exiting an activity with the intent of returning to it. The activity is not marked as complete when exited (see also attempt, resume).

Tracking Model – An information model containing results data for a learner related to objectives and progress on learning activities.

Thread – A collection of related sequenced learning activities.

Wallclock Duration – The cumulative amount of time that a learner spends on an activity, from when the activity starts until it ends (see also Interaction Duration).

G.1 Open Issues

The list of terms and definitions is not complete. [SS.G.01]