SCORM2004 テストスイートによる LMS
検査説明書

第 1.0 版

2006 年 3 月

特定非営利活動法人日本イーラーニングコンソシアム

クレジット表記と再配布のガイドライン

本書は特定非営利活動法人日本イーラーニングコンソシアム（略称：eLC）の著作物である．
本書は経済産業省がスポンサーとなりSCORM（Sharable Content Object Reference Model）の普及促進を目的として作成された．

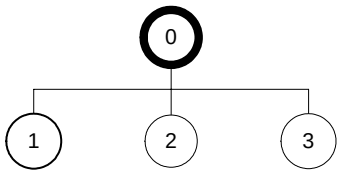
ｅLC（ライセンス提供者）は，他者（ライセンス受領者）に本書のコピー，配布，表示，ハイパーリンクの生成を許可する．代わりに，ライセンス提供者のクレジット（上記アンダーラインの文章）を明記しなければならない．
また，ライセンス受領者はライセンス提供者の許可なく本書を商用利用してはならない．

本書は ADL の TestSuite Conformance Reqts の中から TestSuite1.3.3 で動作しているコンテンツの設定に関する部分を抜き出し、LMS 検査についての補足説明を追記しました。

各テストごとに、テストの目的、使われている設定、動作に関してまとめています。

Test Case: CM-1

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Limit Conditions: Attempt Absolute Duration Limit == P5Y6M4DT12H30M58S
2	Objectives: Primary Objective: Minimum Normalized Measure == 0.8
3	Limit Conditions: Attempt Absolute Duration Limit == P5Y6M4DT12H30M58S Objectives: Primary Objective: Minimum Normalized Measure == 0.7

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 2 for delivery
3.	Process a *Continue* navigation request	Identify Activity 3 for delivery
4.	Process a *Previous* navigation request	Identify Activity 2 for delivery
5.	Process a *Continue* navigation request	Identify Activity 3 for delivery
6.	Process a *Previous* navigation request	Identify Activity 2 for delivery
7.	Process a *Previous* navigation request	Identify Activity 1 for delivery

【目的】
Control Mode Flow がデフォルト値と反対の true の場合のテストを行います。

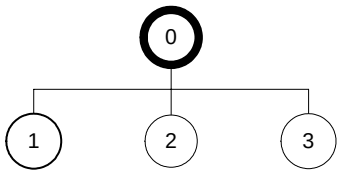
【構造】
Activity0 の Control Mode に Flow=true

【動作】
Activity0 の子 Activity1～3 を Continue、Previous リクエストを用いて前後に移動します。

【補足】
Activity1～3 に関して設定されている項目は使用されていません。

Test Case: CM-2a

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Control Choice == false
1	Default
2	Objectives: Primary Objective: *empty* Sequencing Rules: Precondition Rule: If satisfied, then skip
3	Objectives: Primary Objective: Objective ID == PRIMARYOBJ

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 2 for delivery
3.	Process a *Continue* navigation request	Identify Activity 3 for delivery
4.	Process a *Previous* navigation request	Identify Activity 1 for delivery
5.	Process a *Continue* navigation request	Identify Activity 3 for delivery
6.	Process a *Previous* navigation request	Identify Activity 1 for delivery

【目的】
Skip Precondition Rule が働くことを確認します。また、LMS が Leaf Activity の習得状態の自動設定を実行しているかどうかも合わせて確認します。Continue、Previous リクエストを用いて前後に移動するために、Control Mode Flow がデフォルト値と反対の true に設定されます。

【構造】
Activity0 の Control Mode に Flow=true
Activity2 の Precondition Rule に If satisfied, then skip

【動作】
CM-1 と同様に Activity0 の子 Activity1～3 を Continue、Previous リクエストを用いて前後に移動します。

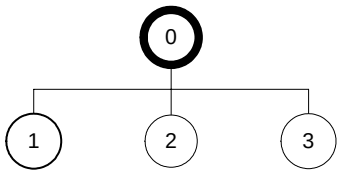
Step2 で最初に Activity2 を試行したとき Activity2 は satisfied になるはずです。これは、Activity2 の SCO からは cmi.success_status が設定されず、Objective Set by Content がデフォルトの false のため、LMS が自動的に Activity2 の習得状態を設定するはずだからです。

Step4 で Activity1 からの Continue、Step5 で Activity3 からの Previous リクエストに対して Activity2 はスキップされます。

【補足】
Activity1～3 に設定されているObjectiveに関する項目は使用されていません。

Test Case: CM-2b

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Control Choice == false
1	Objectives: Primary Objective: *empty* Objective: Objective ID == obj1 Objective: Objective ID == obj2
2	Sequencing Rules: Precondition Rule: If satisfied, then skip Objectives: Primary Objective: Objective ID == PRIMARYOBJ Objective: Objective ID == obj1 Delivery Controls: Objective Set by Content == true
3	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 2 for delivery
3.	Process a *Continue* navigation request	Identify Activity 3 for delivery
4.	Process a *Previous* navigation request	Identify Activity 2 for delivery
5.	Set Activity 2's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 3 for delivery
6.	Process a *Previous* navigation request	Identify Activity 1 for delivery
7.	Process a *Continue* navigation request	Identify Activity 3 for delivery
8.	Process a *Previous* navigation request	Identify Activity 1 for delivery

【目的】
Skip Precondition Rule が働くことを確認します。Skip の条件として SCO から習得状態を設定します。Continue、Previous リクエストを用いて前後に移動するために、Control Mode Flow がデフォルト値と反対の true に設定されます。

【構造】
Activity0 の Control Mode に Flow=true
Activity2 の Precondition Rule に If satisfied, then skip
Activity2 の Delivery Controls の Objective Set by Content = true

【動作】
CM-1 と同様に Activity0 の子 Activity1～3 を Continue、Previous リクエストを用いて前後に移動します。

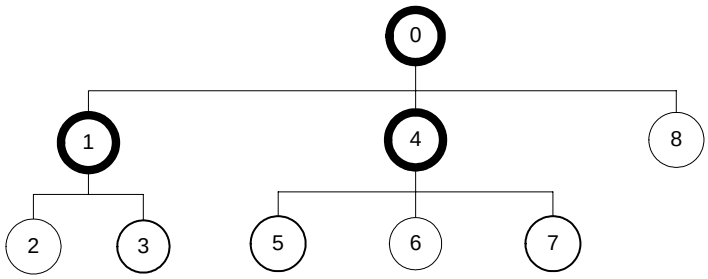
Step2 で最初に Activity2 を試行したときに Activity2 の SCO からはなにも設定されず、かつ、Set by Content に True がセットされているため、Activitiy2 の習得状態は変化しません。このため Step4 では Skip Precondition Rule が成り立たず Activity2 が配信されます。

次に Step5 で Activitiy2 の SCO から cmi.success_status に passed をセットします。このため、Activitiy2 が satisfied となり、Step7 の Activity1 からの Continue、Step8 の Activity3 からの Previous リクエストに対して Activitiy2 がスキップされます。

【補足】
Activity1～3 に設定されている Objective に関する項目は使用されていません。

Test Case: CM-3a

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Control Mode: Flow == true Choice == false
2	Default
3	Default
4	Control Mode: Flow == true Choice == false Forward Only ==true
5	Sequencing Rules: Precondition Rule: If attempted and not completed, then skip
6	Sequencing Rules: Precondition Rule: If attempted and not completed, then skip
7	Sequencing Rules: Precondition Rule: If attempted and not completed, then skip
8	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 2for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Process a *Continue* navigation request	Identify Activity 5 for delivery
4.	Set Activity 5's cmi.completion_status to `completed`; Process a *Continue* navigation request	Identify Activity 6 for delivery

5.	Set Activity 6's cmi.completion_status to `incomplete`; Process a *Continue* navigation request	Identify Activity 7 for delivery
6.	Set Activity 7's cmi.completion_status to `incomplete`; Process a *Continue* navigation request	Identify Activity 8 for delivery
7.	Process a *Previous* navigation request	Identify Activity 5 for delivery

【目的】
Forward Only が true のときの Parevious リクエストの動作および Skip Precondition Rule をテストします。

【構造】
Activity4 に Forward Only =true
Activity5 の Precondition Rule に If attempted and not completed, then skip
Activity6 の Precondition Rule に If attempted and not completed, then skip
Activity7 の Precondition Rule に If attempted and not completed, then skip

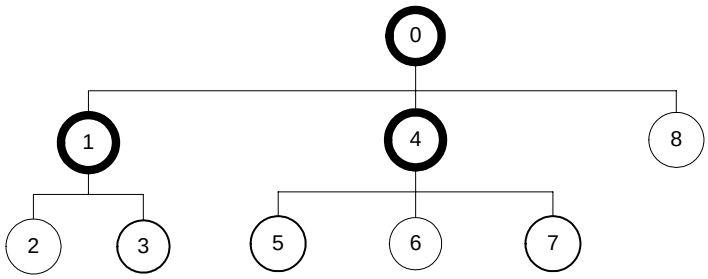
【動作】
Step1 から Step6 で Continue リクエストで Activity2 から Activity8 まで移動します。Activity8 で Previous リクエストをかけたとき、通常は Activity7⇒6⇒5⇒3⇒2 と配信可能な Activity を探しますが、このテストでは Activity4 に Forward Only=true がセットされているため配信対象の検索の順序は Activity5⇒6⇒7⇒3⇒2 となります。

最初の配信対象である Activity5 は Step4 で completion_status に completed がセットされています。一方、Precondition Rule は If attempted and not completed, then skip であるため Skip Precondition Rule は成り立たずに、Step7 では Activity5 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: CM-3b

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Control Model: Flow == true Choice == false
2	Default
3	Default
4	Control Mode: Flow == true Choice == false Forward Only ==true
5	Sequencing Rules: Precondition Rule: If attempted and not completed, then skip
6	Sequencing Rules: Precondition Rule: If attempted and not completed, then skip
7	Sequencing Rules: Precondition Rule: If attempted and not completed, then skip
8	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 2for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Process a *Continue* navigation request	Identify Activity 5 for delivery
4.	Set Activity 5's cmi.completion_status to `incomplete` Process a *Continue* navigation request	Identify Activity 6 for delivery

5.	Set Activity 6's cmi.completion_status to `incomplete`; Process a *Continue* navigation request	Identify Activity 7 for delivery
6.	Set Activity 7's cmi.completion_status to `incomplete`; Process a *Continue* navigation request	Identify Activity 8 for delivery
7.	Process a *Previous* navigation request	Identify Activity 3 for delivery
8	Process a *Previous* navigation request	Identify Activity 2 for delivery

【目的】
Forward Only が true のときの Parevious リクエストの動作および Skip Precondition Rule をテストします。

【構造】
Activity4 に Forward Only =true
Activity5 の Precondition Rule に If attempted and not completed, then skip
Activity6 の Precondition Rule に If attempted and not completed, then skip
Activity7 の Precondition Rule に If attempted and not completed, then skip

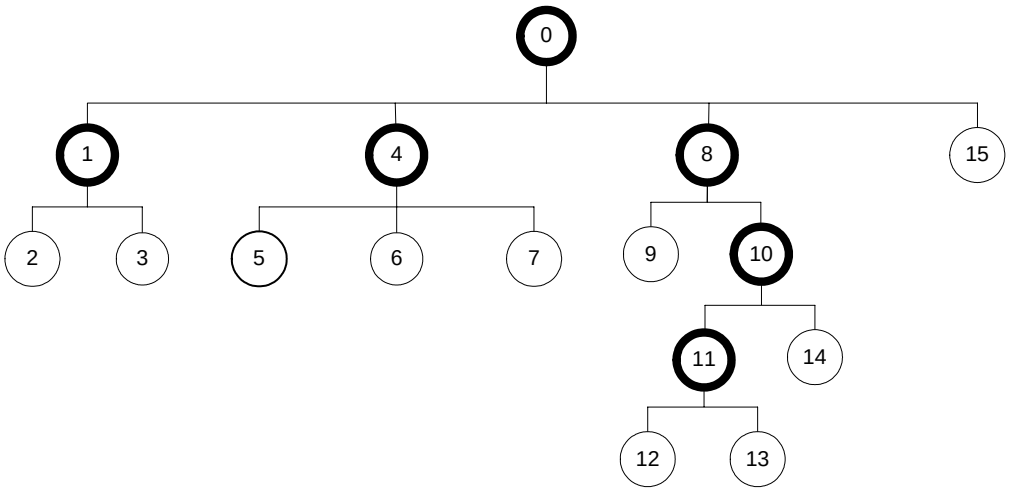
【動作】
Step1 から Step6 で Continue リクエストで Activity2 から Activity8 まで移動します。Activity8 から Previous リクエストをかけたとき、通常は Activity7⇒6⇒5⇒3⇒2 と配信可能な Activity を探しますが、このテストでは Activity4 に Forward Only=true がセットされているため配信対象の検索の順序は Activity5⇒6⇒7⇒3⇒2 となります。

ここで配信対象を順番に見てみると
Activity5 は Step4 で incomplete がセットされているため Precondition Rule により Skip
Activity6 は Step5 で incomplete がセットされているため Precondition Rule により Skip
Activity7 は Step6 で incomplete がセットされているため Precondition Rule により Skip
結果 Step7 での Parevious リクエストでは Activity3 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: CM-4

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Default
1	Default
2	Default
3	Default
4	Default
5	Default
6	Default
7	Default
8	Default
9	Default
10	Default
11	Control Mode: Flow == true
12	Default
13	Default
14	Default
15	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Choice* navigation request for Activity 12	Identify Activity 12 for delivery
2.	Process a *Choice* navigation request for Activity 12	Identify Activity 12 for delivery
3.	Process a *Choice* navigation request for Activity 11	Identify Activity 12 for delivery

4.	Process a *Choice* navigation request for Activity 5	Identify Activity 5 for delivery
5.	Process a *Choice* navigation request for Activity 7	Identify Activity 7 for delivery
6.	Process a *Choice* navigation request for Activity 6	Identify Activity 6 for delivery
7.	Process a *Choice* navigation request for Activity 15	Identify Activity 15 for delivery
8.	Process a *Choice* navigation request for Activity 9	Identify Activity 9 for delivery
9.	Process a *Choice* navigation request for Activity 14	Identify Activity 14 for delivery
10	Process a *Choice* navigation request for Activity 3	Identify Activity 3 for delivery

【目的】
Activity を Choice で移動できるかのテストです。

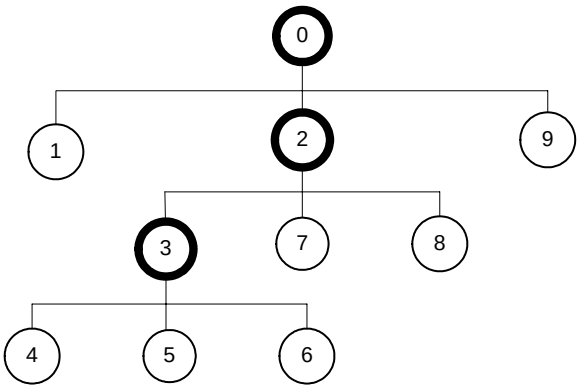
【構造】
なし

【動作】
Activity0 のシーケンシングルールがデフォルトのため Start リクエストで配信されるものはありません。最初に Activity12 を起動する必要があります。あとは Chocie リクエストを正常に処理することで自動的にテストは進みます。

【補足】
Activity11 で Control Mode の Flow が True にセットされており、Flow と Choice が LMS 内で独立に扱われていることを確認しています。

Test Case: CM-5

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true
1	Default
2	Control Mode: Flow == true
3	Control Mode: Flow == true
4	Default
5	Default
6	Default
7	Default
8	Default
9	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Choice* navigation request for Activity 6	Identify Activity 6 for delivery
3.	Process a *Suspend All* navigation request	End Sequencing Session
4.	Process a *Resume All* navigation request	Identify Activity 6 for delivery
5.	Process a *Choice* navigation request for Activity 8	Identify Activity 8 for delivery
6.	Process a *Suspend All* navigation request	End Sequencing Session
7.	Process a *Resume All* navigation request	Identify Activity 8 for delivery

【目的】
中断再開のテストを行います。

【構造】
なし

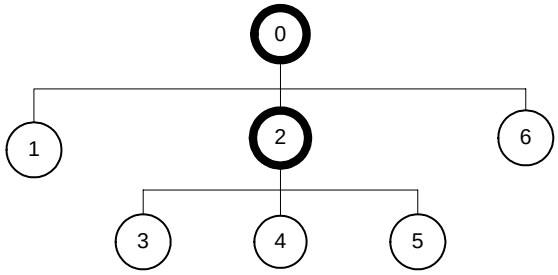
【動作】
Activity6 と Activity8 で中断・再開を行います。前者では LMS の機能を使用しての中断・再開、後者では RunTime を使用しての中断・再開になります。

いずれも、中断時と同じ Activity が再開時に配信される必要があります。

【補足】
中間ノードアクティビティのすべて Control Mode の Frow に True をセットしていますが Activity0 以外は使用されていません。

Test Case: RU-1aa

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule If satisfied, then exit Post Condition Rule: If satisfied, then previous
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 6 for delivery

【目的】
「子 Activity がすべて satisfied なら親 Activity は satisfied になる」というデフォルトロールアップのテストを行います。条件が成り立たない場合の動作をテストします。

【構造】
Activity2 の Exit Rule に If satisfied, then exit
Activity2 の Post Condition Rule に If satisfied, then previous
Activity2 の Rollup Rules を空（デフォルトルール使用のため）

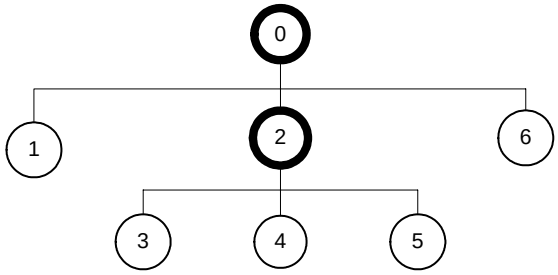
【動作】
Step1〜5 で Activity2 の子 Activity に Satisfied Status を以下のようにセットしながら進みます。
Activity3 の Step3 で success status に passed をセット
Activity4 の Step4 で success status に failed をセット
Activity5 の Step5 で success status に passed をセット

どの Step においても Activity2 が Satisfied になることはありません。このため Activity5 で Continue リクエストが起きたとき、Exit Rule、Previous Post Condition Rule は成り立たず Activity6 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: RU-1ab

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule If satisfied, then exit Post Condition Rule: If satisfied, then previous
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 1 for delivery

【目的】
「子 Activity がすべて satisfied なら親 Activity は satisfied になる」というデフォルトロールアップのテストを行います。条件が成り立たつ場合の動作をテストします。

【構造】
Activity2 の Exit Rule に If satisfied, then exit
Activity2 の Post Condition Rule に If satisfied, then previous
Activity2 の Rollup Rules を空（デフォルトルール使用のため）

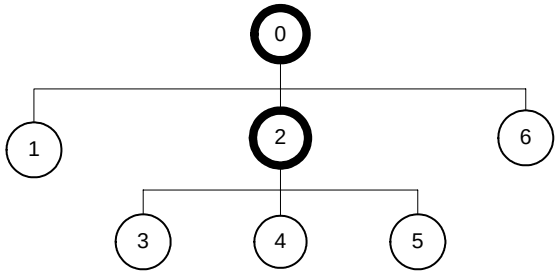
【動作】
Step1〜5 で Activity2 の子 Activity に Satisfied Status を以下のようにセットしながら進みます。
Activity3 の Step3 で success status に passed をセット
Activity4 の Step4 で success status に passed をセット
Activity5 の Step5 で success status に passed をセット

ここで Activity5 から別の Activity に移動しようとすると、Activity2 の子すべてが satisfied のため Activity2 は satisfied となります。Exit Rule が成り立って Previous Post Condition Rule が評価され、条件が成り立って previous リクエストが発生します。このように Step5 の Continue リクエストの代わりに previous リクエストが実行されて Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: RU-1ba

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Control Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If completed, then exit Post Condition Rule: If completed, then previous
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.completion_status to `completed`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.completion_status to `completed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.completion_status to `incomplete`; Process a *Continue* navigation request	Identify Activity 6 for delivery

【目的】
「子 Activity がすべて completed なら親 Activity は completed になる」というデフォルトロールアップのテストを行います。条件が成り立たない場合の動作をテストします。

【構造】
Activity2 の Exit Rule に If completed, then exit
Activity2 の Post Condition Rule に If completed, then previous
Activity2 の Rollup Rules を空（デフォルトルール使用のため）

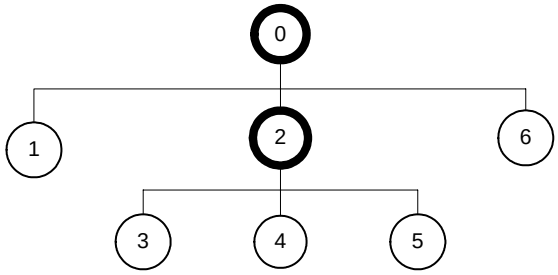
【動作】
Step1〜5 で Activity2 の子 Activity に Progress Status を以下のようにセットしながら進みます。
Activity3 の Step3 で success status に completed をセット
Activity4 の Step4 で success status に incomplete をセット
Activity5 の Step5 で success status に completed をセット

どの Step においても Activity2 が completed になることはありません。このため Activity5 で Continue リクエストが起きたとき、、Exit Rule、Previous Post Condition Rule は成り立たず Activity6 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: RU-1bb

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If completed, then exit Post Condition Rule: If completed, then previous
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.completion_status to `completed`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.completion_status to `completed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.completion_status to `completed`; Process a *Continue* navigation request	Identify Activity 1 for delivery

【目的】
「子 Activity がすべて completed なら親 Activity は completed になる」というデフォルトロールアップのテストを行います。条件が成り立たつ場合の動作をテストします。

【構造】
Activity2 の Exit Rule に If completed, then exit
Activity2 の Post Condition Rule に If completed, then previous
Activity2 の Rollup Rules を空（デフォルトルール使用のため）

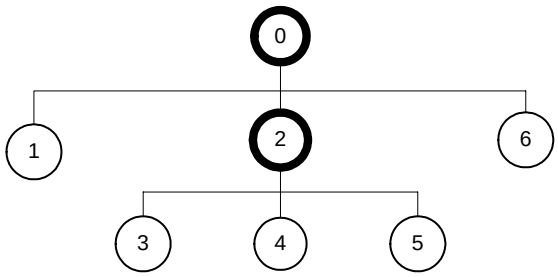
【動作】
Step1〜5 で Activity2 の子 Activity に Progress Status を以下のようにセットしながら進みます。
Activity3 の Step3 で completion status に completed をセット
Activity4 の Step4 で completion status に completed をセット
Activity5 の Step5 で completion status に completed をセット

ここで Activity5 から別の Activity に移動しようとすると、Activity2 の子すべてが completed のため Activity2 は completed となります。Exit Rule が成り立って Previous Post Condition Rule が評価され、条件が成り立って previous リクエストが発生します。このように Step5 の Continue リクエストの代わりに previous リクエストが実行されて Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: RU-2a

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If completed, then exit Post Condition Rule: If completed, then previous Rollup Rules: Completed if any satisfied
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 1 for delivery

【目的】
「任意の子 Activity が satisfied なら親 Activity は completed になる」という Rollup Rules による動作をテストします。

【構造】
Activity2 の Exit Rule に If completed, then exit
Activity2 の Post Condition Rule に If completed, then previous
Activity2 の Rollup Rules に Completed if any satisfied

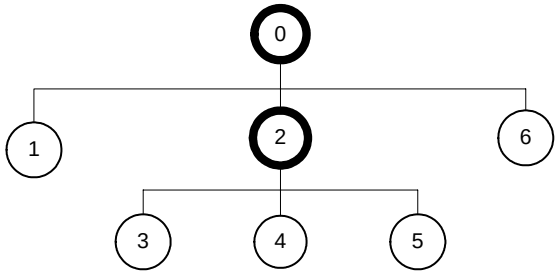
【動作】
Step3 で Activity3 の success status に failed をセットします。この時点では、Activity2 の子 Activity に satisfied のものが存在していないため Continue リクエストで Activity4 が配信されます。

Step4 で Activity3 の success status に passed をセットします。これによって Activity3 が satisfied になり、Activity2 の Rollup Rule が成り立って Activity2 が Completed になります。そのため、Continue リクエストに対して、Exit Rule が成り立って Previous Post Condition Rule が評価され、条件が成り立って previous リクエストが発生し、Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: RU-2b

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If satisfied, then exit Post Condition Rule: If satisfied, then continue Rollup Rules: Satisfied if any completed
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.completion_status to `incomplete`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.completion_status to `completed`; Process a *Previous* navigation request	Identify Activity 6 for delivery

【目的】

「任意の子 Activity が completed なら親 Activity は satisfied になる」という Rollup Rules による動作をテストします。

【構造】
Activity2 の Exit Rule に If satisfied, then exit
Activity2 の Post Condition Rule に If satisfied, then continue
Activity2 の Rollup Rules に Satisfied if any completed

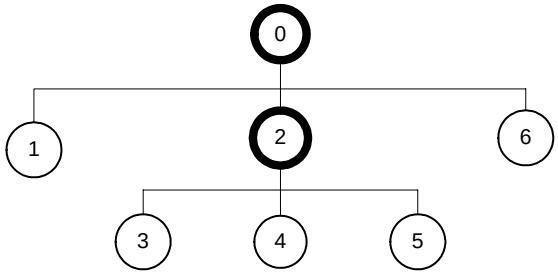
【動作】
Step3 で Activity3 の completion_status に incomplete をセットしてます。この時点では、Activity2 の子 Activity に completed のものが存在していないため Continue リクエストで Activity4 が配信されます。

Step4 で Activity3 の completion_status に completed をセットします。これによって Activity3 が completed になり、Activity2 の Rollup Rule が成り立って Activity2 が satisfied になります。そのため Continue リクエストに対して、Exit Rule が成り立って Previous Post Condition Rule が評価され、条件が成り立って previous リクエストが発生し、Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: RU-3a

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If completed then exit Post Condition Rule: If completed, then previous Rollup Rules: Completed if at least 1 attempted and not completed
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.completion_status to `completed`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.completion_status to `incomplete`; Process a *Previous* navigation request	Identify Activity 1 for delivery

【目的】
「少なくとも１つの子 Activity が試行されかつ not completed なら親 Activity は Completed になる」という Rollup Rules による動作をテストします。

【構造】
Activity2 の Exit Rule に If completed, then exit
Activity2 の Post Condition Rule に If completed, then previous
Activity2 の Rollup Rules に Completed if at least 1 attempted and not completed

【動作】
Step3 で Activity3 の completion_status に completed をセットしています。この時点では Activity2 の Rollup Rules は成り立たないため Continue リクエストで Activity4 が配信されます。

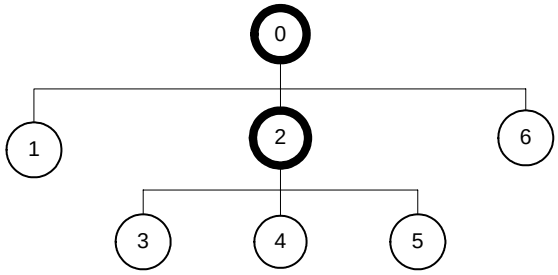
Step4 でも Activity4 の SCO からなにもセットされないため、Activity4 は自動的に completed になります。従って Activity2 の Rollup Rules は成り立たず Continue リクエストでは Activity5 が配信されます。

Step5 で Activity5 の completion_status に incomplete をセットします。これによって Activity5 が attempted かつ not completed になり、Activity2 の Rollup Rules が成り立って Activity2 が completed になります。そのため Continue リクエストに対して、Exit Rule が成り立って Previous Post Condition Rule が評価され、条件が成り立って previous リクエストが発生し、Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: RU-3b

Activity Tree Structure



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If satisfied, then exit Post Condition Rule: If satisfied, then previous Rollup Rules: Satisfied if at least 1 attempted and not satisfied
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 1 for delivery

【目的】
「少なくとも１つの子 Activity が試行されかつ not satisfied なら親 Activity は satisfied になる」という Rollup Rules による動作をテストします。

【構造】
Activity2 の Exit Rule に If satisfied, then exit
Activity2 の Post Condition Rule に If satisfied, then previous
Activity2 の Rollup Rules に Satisfied if at least 1 attempted and not satisfied

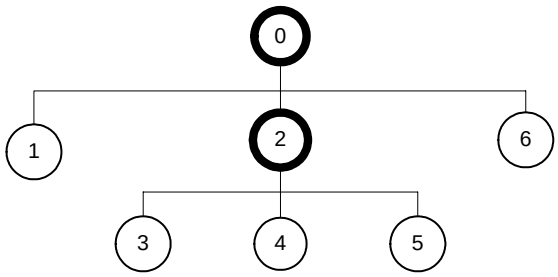
【動作】
Step3 では Activity3 の SCO からなにもセットされないため、Activity3 は自動的に satisfied になります。この時点では Activity2 の Rollup Rules は成り立たないため Continue リクエストで Activity4 が配信されます。

Step4 で Activity4 の success_status に failed をセットします。これによって Activity5 が attempted かつ not satisfied になり、Activity2 が Rollup Rules が成り立って Activity2 が satisfied になります。そのため Continue リクエストに対して、Exit Rule が成り立って Previous Post Condition Rule が評価され、条件が成り立って previous リクエストが発生し、Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: RU-4aa

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If satisfied then exit Post Condition Rule: If satisfied, then previous Rollup Rules: Satisfied if at least 50% satisfied or objective measure known
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.completion_status to `incomplete`; Set Activity 3's cmi.score.scaled to `0.75`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.completion_status to `completed`; Set Activity 4's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.success_status to `failed`;	Identify Activity 6 for delivery

	Process a *Continue* navigation request	

【目的】
「少なくとも 50%の子 Activity が satisfied もしくは objective measure known なら親 Activity は satisfied になる」という Rollup Rules のテストを行います。条件が成り立たない場合の動作をテストします。

【構造】
Activity2 の Exit Rule に If satisfied, then exit
Activity2 の Post Condition Rule に If satisfied, then previous
Activity2 の Rollup Rules に Satisfied if at least 50% satisfied or objective measure known

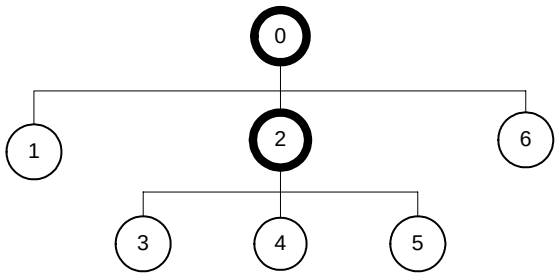
【動作】
Step3 では Activity3 の SCO から score.scaled に 0.75 がセットされているため、Activity3 の objective measure known は True になります。しかし 3 個ある子 Activity のうちの 1 つの子 Activity の objective measure known が True になっただけなので、Activity2 の Rollup Rules は成り立たず Continue リクエストで Activity4 が配信されます。

Step4、Step5 では Activity4、Activity5 の success_status に failed をセットします。そのため、Activity2 の Rollup Rules は成り立たず Continue リクエストが有効になります。結局 Step5 の Continue リクエストに対して Activity6 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。
また SCO から completion_status がセットされていますが、こちらは使われていません。

Test Case: RU-4ab

Activity Tree Structure;



Sequencing Information;

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If satisfied then exit Post Condition Rule: If satisfied, then previous Rollup Rules: Satisfied if at least 50% satisfied or objective measure known
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.completion_status to `incomplete`; Set Activity 3's cmi.score.scaled to `0.75`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.completion_status to `completed`; Set Activity 4's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.success_status to `passed`;	Identify Activity 1 for delivery

	Process a *Previous* navigation request	

【目的】
「少なくとも 50%の子 Activity が satisfied もしくは objective measure known なら親 Activity は satisfied になる」という Rollup Rules のテストを行います。条件が成り立たつ場合の動作をテストします。

【構造】
Activity2 の Exit Rule に If satisfied, then exit
Activity2 の Post Condition Rule に If satisfied, then previous
Activity2 の Rollup Rules に Satisfied if at least 50% satisfied or objective measure known

【動作】
Step3 では Activity3 の SCO から score.scaled に 0.75 がセットされているため、Activity3 の objective measure known は True になります。しかし 3 個ある子 Activity のうちの 1 つの子 Activity の objective measure known が True になっただけなので、Activity2 の Rollup Rules は成り立たず Continue リクエストで Activity4 が配信されます。

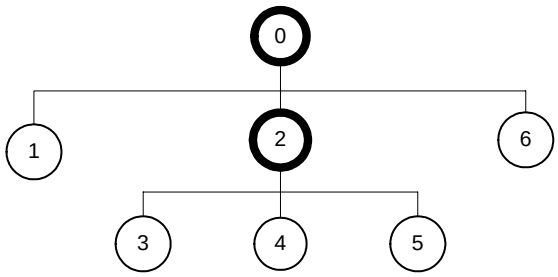
Step4 では Activity4 の success_status に failed をセットします。そのため、Activity2 の Rollup Rules は成り立たず Activity が配信されます。

Step5 では Activity5 の success_status に passed をセットします。これによって Activity2 の子 Activity の 50%以上（Activity3 と Activity5）が satisfied もしくは objective measure known となって Rollup Rules が成り立ち、Activity2 が satisfied になります。そのため Previous リクエストに対して、Exit Rule が成り立って Previous Post Condition Rule が評価され、条件が成り立って Activity2 を起点とする previous リクエストが発生し、Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。
また SCO から completion_status がセットされていますが、こちらは使われていません。

Test Case: RU-4ba

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If completed, then exit Post Condition Rule: If completed, then previous Rollup Rules: Completed if at least 50% completed or objective measure known
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a Start navigation request	Identify Activity 1 for delivery
2.	Process a Continue navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.completion_status to `incomplete`; Process a Continue navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.success_status to `failed`; Process a Continue navigation request	Identify Activity 5 for delivery
5.	Set Activity 5s cmi.completion_status to `incomplete`; Process a Continue navigation request	Identify Activity 6 for delivery

【目的】
「少なくとも 50%の子 Activity が completed もしくは objective measure known なら親 Activity は completed になる」という Rollup Rules のテストを行います。条件が成り立たない場合の動作をテストします。

【構造】
Activity2 の Exit Rule に If completed, then exit
Activity2 の Post Condition Rule に If completed, then previous
Activity2 の Rollup Rules に Completed if at least 50% completed or objective measure known

【動作】
Step3 では Activity3 の SCO から completion_status に incomplete をセットします。この時点で Activity2 のロールアップルールは成り立ちません。

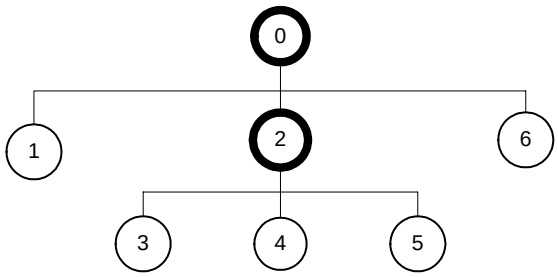
Step4 では completion_status になにもセットしていないため Activity4 は自動的に completed になります。しかし、3 個ある子 Activity のうち 1 つの子 Activity が completed になっただけで、まだロールアップルールは成り立ちません。

Step5 では Activity5 の completion_status に incomplete をセットします。したがって、Activity2 のロールアップルールは成り立ちません。そのため、Continue リクエストが有効となり Activity6 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。
また SCO から success_status がセットされていますが、こちらは使われていません。

Test Case: RU-4bb

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If completed, then exit Post Condition Rule: If completed, then previous Rollup Rules: Completed if at least 50% completed or objective measure known
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.completion_status to `incomplete`; Set Activity 3's cmi.score.scaled to `0.75`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.success_status to `failed`; Set Activity 4's cmi.score.scaled to `0.25`; Process a *Continue* navigation request	Identify Activity 1 for delivery

【目的】
「少なくとも 50%の子 Activity が completed もしくは objective measure known なら親 Activity は completed になる」という Rollup Rules のテストを行います。条件が成り立たつ場合の動作をテストします。

【構造】
Activity2 の Exit Rule に If completed, then exit
Activity2 の Post Condition Rule に If completed, then previous
Activity2 の Rollup Rules に Completed if at least 50% completed or objective measure known

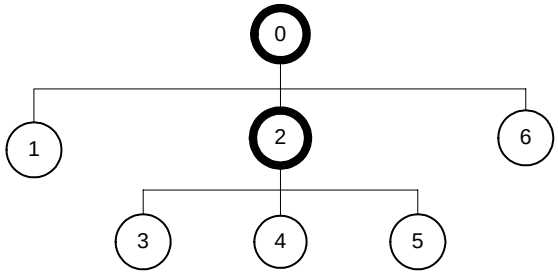
【動作】
Step3 では Activity3 の SCO から score.scaled に 0.75 がセットされているため、Activity3 の objective measure known は True になります。しかし 3 個ある子 Activity のうちの 1 つの子 Activity の objective measure known が True になっただけなので、Activity2 の Rollup Rules は成り立たず Continue リクエストで Activity4 が配信されます。

Step4 では Activity4 の SCO から score. scaled に 0.25 がセットされているため、Activity4 の objective measure known は True になります。ここで Activity2 の子 Activity の 50%以上（Activity3 と Activity4）が completed もしくは objective measure known となって Rollup Rules が成り立ち、Activity2 が completed になります。そのため continue リクエストに対して、Exit Rule が成り立って Previous Post Condition Rule が評価され、条件が成り立って previous リクエストが発生し、Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。
また SCO から success_status にステータスがセットされていますが、こちらは使われていません。

Test Case: RU-4bc

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If completed, then exit Post Condition Rule: If completed, then previous Rollup Rules: Completed if at least 50% completed or objective measure known
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.completion_status to `incomplete`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5s cmi.completion_status to `incomplete`; Process a *Continue* navigation request	Identify Activity 6 for delivery
6.	Process a *Previous* navigation request	Identify Activity 5 for delivery

7.	Process a *Previous* navigation request	Identify Activity 4 for delivery
8.	Set Activity 4's cmi.score.scaled to `0.25`; Process a *Previous* navigation request	Identify Activity 1 for delivery

【目的】
「少なくとも 50%の子 Activity が completed もしくは objective measure known なら親 Activity は completed になる」という Rollup Rules のテストを行います。条件が成り立たつ場合と成り立たない場合の動作をテストします。

【構造】
Activity2 の Exit Rule に If completed, then exit
Activity2 の Post Condition Rule に If completed, then previous
Activity2 の Rollup Rules に Completed if at least 50% completed or objective measure known

【動作】
Step1～5 で Activity3, 4, 5 に completion_status をセットしながら Continue リクエストで進みます。ここまでは Activity2 のロールアップルールは成り立たず、Step5 の Continue リクエストで Activity6 が配信されます。

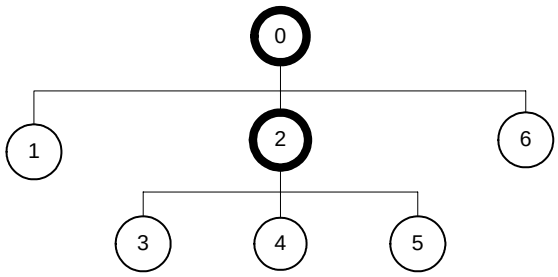
Step7 では completion_status になにもセットしていないため Activity5 は自動的に completed になります。しかし、3 個ある子 Activity のうち 1 つの子 Activity が completed になっただけで、まだロールアップルールは成り立ちません。

Step8 では Activity4 の SCO から score. scaled に 0.25 がセットされているため、Activity4 の objective measure known は True になります。ここで Activity2 の子 Activity の 50%以上（Activity4 と Activity5）が completed もしくは objective measure known となって Rollup Rules が成り立ち、Activity2 が completed になります。そのため previous リクエストに対して、Exit Rule が成り立って Previous Post Condition Rule が評価され、条件が成り立って Activity2 を起点とする previous リクエストが発生し、Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。
また SCO から success_status がセットされていますが、こちらは使われていません。

Test Case: RU-4bd

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If completed, then exit Post Condition Rule: If completed, then previous Rollup Rules: Completed if at least 50% completed or objective measure known
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.completion_status to `incomplete`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5s cmi.completion_status to `incomplete`; Process a *Continue* navigation request	Identify Activity 6 for delivery
6.	Process a *Previous* navigation request	Identify Activity 5 for delivery

7.	Set Activity 5s cmi.completion_status to `incomplete`; Process a *Previous* navigation request	Identify Activity 4 for delivery
8.	Set Activity 4's cmi.score.scaled to `0.8`; Process a *Continue* navigation request	Identify Activity 5 for delivery
9.	Set Activity 5's cmi.score.scaled to `0.5`; Process a *Continue* navigation request	Identify Activity 1 for delivery

【目的】
「少なくとも 50%の子 Activity が completed もしくは objective measure known なら親 Activity は completed になる」という Rollup Rules のテストを行います。条件が成り立たつ場合と成り立たない場合の動作をテストします。

【構造】
Activity2 の Exit Rule に If completed, then exit
Activity2 の Post Condition Rule に If completed, then previous
Activity2 の Rollup Rules に Completed if at least 50% completed or objective measure known

【動作】
Step1～5 で Activity3, 4, 5 に completion_status をセットしながら Continue リクエストで進みます。ここまでは Activity2 のロールアップルールは成り立たず、Step5 の Continue リクエストで Activity6 が配信されます。

Step7 では Activity5 の completion_status に incomplete をセットします。この時点でロールアップルールは成り立ちません。

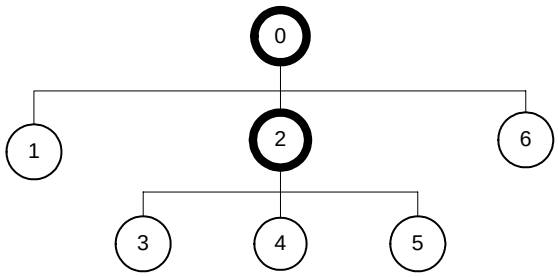
Step8 で Activity4 の SCO から score.scaled に 0.8 がセットされているため、Activity4 の objective measure known は True になります。しかし、3 個ある子 Activity のうち 1 つの子 Activity の objective measure known が true になっただけで、まだロールアップルールは成り立ちません。

Step9 で Activity5 の SCO から score.scaled に 0.5 がセットされているため、Activity5 の objective measure known は true になります。ここで Activity2 の子 Activity の 50%以上（Activity4 と Activity5）が completed もしくは objective measure known となって Rollup Rules が成り立ち、Activity2 が completed になります。そのため previous リクエストに対して、Exit Rule が成り立って Previous Post Condition Rule が評価され、条件が成り立って Activity2 を起点とする previous リクエストが発生し、Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。
また SCO から success_status にステータスがセットされていますが、こちらは使われていません。

Test Case: RU-5a

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If completed, then exit Post Condition Rule: If completed, then previous Rollup Rules: Completed if none satisfied
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 1 for delivery

【目的】
「子 Activity に satisfied がなければ親 Activity は completed になる」という Rollup Rules のテストを行います。条件が成り立つ場合の動作をテストします。

【構造】
Activity2 の Exit Rule に If completed, then exit
Activity2 の Post Condition Rule に If completed, then previous
Activity2 の Rollup Rules に Completed if none satisfied

【動作】
Step1 から Continue リクエストで進みます。

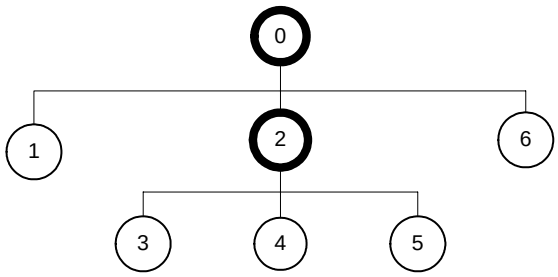
Step3, 4 で Activity3, 4 の success_status が failed がセットされています。しかし、Activity5 の success_status が unknown のため、Activity2 のロールアップルールは成り立ちません。

Step5 で Activity5 の success_status が failed がセットされて Activity2 のロールアップルールが成り立ち completed になります。そのため continue リクエストに対して、Exit Rule が成り立って Previous Post Condition Rule が評価され、条件が成り立って Activity2 を起点とする previous リクエストが発生し、Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: RU-5b

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If satisfied, then exit Post Condition Rule: If satisfied, then continue Rollup Rule: Satisfied if none completed
3	Default
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.completion_status to `incomplete`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.completion_status to `completed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.completion_status to `incomplete`; Process a *Previous* navigation request	Identify Activity 4 for delivery
6.	Set Activity 4's cmi.completion_status to `incomplete`;	Identify Activity 6 for delivery

	Process a *Previous* navigation request	

【目的】
「子 Activity に completed がなければ親 Activity は satisfied になる」という Rollup Rules のテストを行います。

【構造】
Activity2 の Exit Rule に If satisfied, then exit
Activity2 の Post Condition Rule に If satisfied, then continue
Activity2 の Rollup Rules に Satisfied if none completed

【動作】
Step1 から Continue リクエストで進みます。

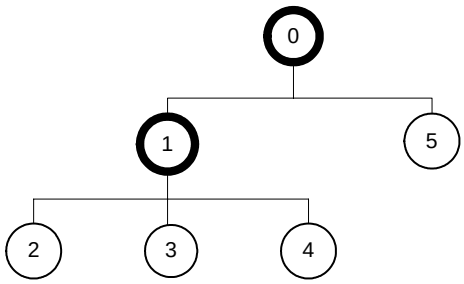
Step3, 4, 5 で Activity3, 4, 5 に incomplete, completed, incomplete とセットします。Activity4 が completed のため、Activity2 のロールアップルールは成り立ちません。

Step5 で Activity4 にもどり、Step6 で Activity4 に incomplete をセットすることでロールアップルールは成り立ちます。previous リクエストに対して、Exit Rule が成り立って Continue Post Condition Rule が評価され、条件が成り立って continue リクエストが発生し、Activity6 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: RU-6a

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Control Mode: Flow == true Choice == false Forward Only == true Use Current Attempt Objective Information == false Sequencing Rules: Exit Rule: If satisfied, then exit Post Condition Rule: If satisfied, then continue
2	Default
3	Default
4	Default
5	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 2 for delivery
2.	Set Activity 2's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Process a *Previous* navigation request	Identify Activity 2 for delivery
6.	Set Activity 2's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 5 for delivery

【目的】
Use Current Attempt Objective Information が false の時の、デフォルトロールアップルールによる動作をテストします。

【構造】
Activity1 の Use Current Attempt Objective Information を false
Activity1 の Exit Rule に If satisfied, then exit
Activity1 の Post Condition Rule に If satisfied, then continue
Activity1 の Rollup Rules を空（デフォルトルール使用のため）

【動作】
Step2, 3, 4 で Activity2, 3, 4 に failed, passed, passed をセットします。Activity2 が failed のためデフォルトロールアップルールは成り立たず、Step4 の Continue リクエストで Activity5 が配信されます。

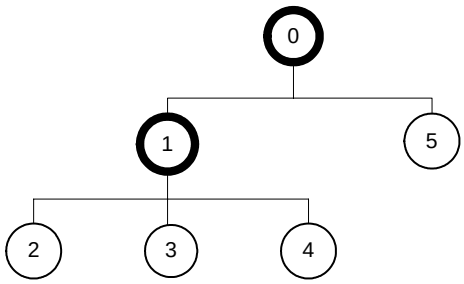
Step5 の Previous リクエストで、Activity1 の Forward Only が true のため、Activity4 でなく Activity2 に戻ります。

Step6 で Activity2 に passed をセットします。Activity1 の Use Current Attempt Objective Information が false であるため、前のアテンプト（Step3, 4）で Activity3, 4 にセットされた passed の Status が有効となり、Activity2, 3, 4 がすべて Satisfied と評価されて、デフォルトロールアップルールが成り立たちます。そのため continue リクエストに対して、Exit Rule が成り立って Continue Post Condition Rule が評価され、条件が成り立って Activity1 を起点とする continue リクエストが発生し、Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。
Activity1 の Forward Only に True がセットされていますが、これは配信アクティビティの検索順序を制御するためです。

Test Case: RU-6b

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Control Mode: Flow == true Choice == false Use Current Attempt Progress Information == false Sequencing Rules: Exit Rule: If satisfied, then exit Post Condition Rule: If satisfied, then continue Rollup Rules: Satisfied if at least 2 completed
2	Default
3	Default
4	Default
5	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 2 for delivery
2.	Set Activity 2's cmi.completion_status to `incomplete`; Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.completion_status to `completed`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.completion_status to `incomplete`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Process a *Previous* navigation request	Identify Activity 4 for delivery
6.	Set Activity 4's cmi.completion_status to `completed`;	Identify Activity 5 for delivery

	Process a *previous* navigation request	

【目的】
Use Current Attempt Objective Information が false の時のロールアップルールによる動作をテストします。「すくなくとも２つの子 Activity が completed なら親 Activity は satisfied になる」というロールアップルールを用います。

【構造】
Activity1 の Use Current Attempt Objective Information を false
Activity1 の Exit Rule に If satisfied, then exit
Activity1 の Post Condition Rule に If satisfied, then continue
Activity1 の Rollup Rules を Satisfied if at least 2 completed

【動作】
Step2, 3, 4 で Activity2, 3, 4 に incomplete, completed, incompleted をセットします。二つの Activity が incomplete のためデフォルトロールアップルールは成り立たず、Step4 の Continue リクエストで Activity5 が配信されます。

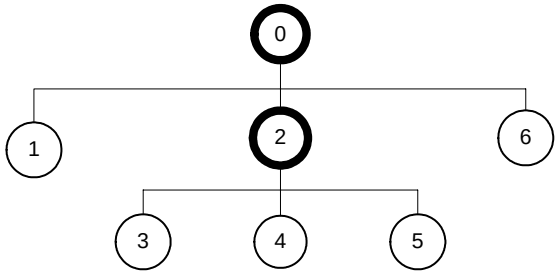
Step5 の Previous リクエストで Activity4 に戻ります。

Step6 で Activity4 に completed をセットします。Activity1 の Use Current Attempt Objective Information が false であるため、前のアテンプト（Step3）で Activity3 にセットされた completed の Status が有効となり、Activity3, 4 が completed と評価されて、ロールアップルールが成り立たちます。そのため previous リクエストに対して、Exit Rule が成り立って Continue Post Condition Rule が評価され、条件が成り立って Activity1 を起点とする continue リクエストが発生し、Activity5 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: RU-7a

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If completed, then exit Post Condition Rule: If not satisfied, then retry
3	Limit Conditions: Attempt limit == 1 Sequencing Rules: Precondition Rule: If attempt limit exceeded, then skip Rollup Considerations: Required for Satisfied if not skipped Required for Not Satisfied if not skipped Rollup Rules: Rollup Progress Completion == false
4	Sequencing Rules: Post Condition Rule: If always, then skip; Rollup Considerations: Required for Completion if attempted Required for Satisfied if attempted Required for Not Satisfied if attempted
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
4.	Set Activity 5's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
6.	Set Activity 5's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 6 for delivery

【目的】

Rollup Considerations が設定されたの時のロールアップルールによる動作をテストします。デフォルト satisfied ロールアップルール「すべての子 Activity が satisfied なら親 Activity は satisfied になる」、デフォルト not satisfied ロールアップルール「すべての子 Activity が試行されるか not satisfied なら親 Activity は not satisfied になる」、デフォルト completed ロールアップルール「すべての子 Activity が completed なら親 Activity は completed になる」という 3 つのデフォルトロールアップルールを用います。

【構造】

Activity2 の Exit Rule を If completed, then exit
Activity2 の Post Condition Rule に If not satisfied, then retry
Activity3 の Limit Conditions の Attempt limit の 1
Activity3 の Precondition Rule を If attempt limit exceeded, then skip
Activity3 の Post Condition Rule を If not satisfied, then retry
Activity3 の Rollup Considerations を Required for Satisfied if not skipped
Activity3 の Rollup Considerations を Required for Not Satisfied if not skipped
Activity3 の Rollup Rules の Rollup Progress Completion を false
Activity4 の Precondition Rule を If always, then skip
Activity4 の Rollup Considerations を Required for Completion if attempted
Activity4 の Rollup Considerations を Required for Satisfied if attempted
Activity4 の Rollup Considerations を Required for Not Satisfied if attempted

【動作】

Step3 で Activity3 の SCO から success_status に failed をセットします。この時点で Activity2 のデフォルト completed ロールアップルールを評価すると以下のようになります。まず、Activity3 の完了状態は、Rollup Progress Completion が false なのでルールの評価対象に含まれません。Activity4 の完了状態も Rollup Considerations が Required for Completion if attempted で、まだ Activity4 は attempted ではないのでルールの評価対象に含まれません。Activity5 はルールの評価対象になります。Activity5 のステータスは unknwon のためルールは成立しません。同様に、デフォルト satisfied ロールアップルール、デフォルト not satisfied ロールアップルールも、Activity5 が unknwon のため成立しません。ここで Continue リクエストにより配信 Activity を探します。Activity4 の Precondition Rule は If always, then skip のため Skip されて Activity5 が配信されます。

Step4 で Activity5 の SCO から success_status に failed をセットします。progress_status にはなにもセットしないため completed となります。この時点で Activity2 のデフォルト completed ロールアップルールを評価すると、条件は前と同じのため、Activity5 のみが評価対象となります。Activity5 が completed のため Activity2 のデフォルト completed ロールアップルールが成り立たち completed になります。そのため Continue リクエストに対して、Exit Rule が成り立って Retry Post Condition Rule が評価されます。Activity4 は not attempted のため Rollup Considerations によって satisfied ルール、not satisfied ルールの評価対象とならず、Activity3 は attempted、Activity5 は success status が failed のため、デフォルト not satisfied ロールアップルールが成り立って Activity2 は not satisfied となり Activity2 を起点とする Retry リクエストが発生します。ここで配信対象をみていくと Activity3 の Attempt limit が 1 で Precondition Rule が If attempt limit exceeded, then skip であることから Skip されて、Activity4 の Precondition Rule も If always, then skip のため Skip されて Activity5 が配信されます。

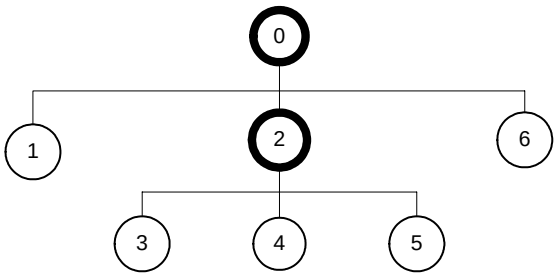
Step5 でも Activity5 の SCO から success_status に failed をセットします。Progress status は Step4 と同じ状態であるため同じように Exit Rule が成り立って Retry Post Condition Rule が評価されます。今度は、Activity3 は skipped、Activity4 は not attempted のため、Rollup Considerations によってどちらも satisfied ルール、not satisfied ルールの評価対象となりません。Activity5 のみが評価対象となり、前と同じように条件が成り立って Activity2 を起点とする Retry リクエストが発生し、配信対象も同様に Activity3、Activity4 が Skip されて Activity5 が配信されます。

Step6 は Activity5 の SCO から success_status に passed をセットします。Progree status は Step4 と同じ状態であるため同じように Exit Rule が成り立って Retry Post Condition Rule が評価されます。ここでは Activity5 の success status が passed のためデフォルト satisfied ロールアップルールがが成り立ち Activity2 は satisfied になります。そのため Retry Post Condition Rule が評価されますが成立せずに Continue リクエストのまま Activity6 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: RU-7b

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If completed, then exit Post Condition Rule: If not satisfied, then retry
3	Limit Conditions: Attempt limit == 1 Sequencing Rules: Precondition Rule: If attempt limit exceeded, then skip Rollup Considerations: Required for Satisfied if not skipped Rollup Rules: Rollup Progress Completion == false
4	Sequencing Rules: Precondition Rule: If always, then skip Rollup Considerations: Required for Completion if attempted Required for Satisfied if attempted
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
4.	Set Activity 5's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 6 for delivery

【目的】
Rollup Considerations が設定されたの時のロールアップルールによる動作をテストします。デフォルト satisfied ロールアップルール「すべての子 Activity が satisfied なら親 Activity は satisfied になる」、デフォルト completed ロールアップルール「すべての子 Activity が completed なら親 Activity は completed になる」という 3 つのデフォルトロールアップルールを用います。

【構造】
Activity2 の Exit Rule を If completed, then exit
Activity2 の Post Condition Rule に If not satisfied, then retry
Activity3 の Limit Conditions の Attempt limit の 1
Activity3 の Precondition Rule を If attempt limit exceeded, then skip
Activity3 の Rollup Consideration を Required for Satisfied if not skipped
Activity3 の Rollup Rules の Rollup Progress Completion を false
Activity4 の Precondition Rule を If always, then skip
Activity4 の Rollup Considerations を Required for Completion if attempted
Activity4 の Rollup Considerations を Required for Satisfied if attempted

【動作】
Step3 で Activity3 の SCO から success_status に passed をセットします。この時点で Activity2 のデフォルト completed ロールアップルールを評価すると以下のようになります。まず、Activity3 の完了状態は、Rollup Progress Completion が false なのでルールの評価対象に含まれません。Activity4 の完了状態も Rollup Considerations が Required for Completion if attempted で、まだ Activity4 は attempted ではないのでルールの評価対象に含まれません。Activity5 はルールの評価対象になります。Activity5 のステータスは unknwon のためルールは成立しません。同様に、デフォルト satisfied ロールアップルール、デフォルト not satisfied ロールアップルールも、Activity5 が unknwon のため成立しません。ここで Continue リクエストにより配信 Activity を探します。Activity4 の Precondition Rule は If always, then skip のため Skip されて Activity5 が配信されます。

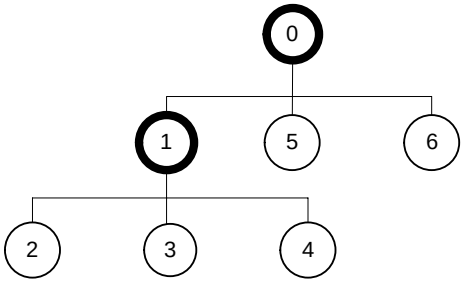
Step4 で Activity5 の SCO から success_status に passed をセットします。progress_status にはなにもセットしないため completed となります。この時点で Activity2 のデフォルト completed ロールアップルールを評価すると、条件は前と同じのため、Activity5 のみが評価対象となります。Activity5 が completed のため Activity2 のデフォルト completed ロールアップルールが成り立たち completed になります。そのため Continue リクエストに対して、Exit Rule が成り立って Retry Post Condition Rule が評価されます。Activity4 は not attempted のため Rollup Considerations によって satisfied ルールの評価対象とならず、Activity3、Activity5 は success status が passed のため、デフォルト satisfied ロールアップルールが成り立って Activity2 は

satisfied となり Retry Post Condition Rule 条件が成り立たず、Continue リクエストのまま Activity6 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: RU-7c

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If completed, then exit Post Condition Rule: If not satisfied, then retry
1	Control Mode: Flow == true Choice == false Limit Conditions: Attempt limit == 2 Sequencing Rules: Precondition Rule: If attempt limit exceeded, then skip Rollup Considerations: Required for Satisfied if not skipped Rollup Rules: Rollup Progress Completion == false
2	Default
3	Default
4	Default
5	Default
6	Rollup Rules: Rollup Progress Completion == false Rollup Objective Satisfied == false

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 2 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.success_status to `passed`; Set Activity 5's cmi.progress_status to `completed`; Process a *Continue* navigation request	Identify Activity 2 for delivery
6.	Set Activity 2's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 3 for delivery
7.	Process a *Continue* navigation request	Identify Activity 4 for delivery
8.	Process a *Continue* navigation request	Identify Activity 5 for delivery
9.	Set Activity 5's cmi.success_status to `failed`; Set Activity 5's cmi.progress_status to `completed`; Process a *Continue* navigation request	Identify Activity 5 for delivery

【目的】
Rollup Considerations が設定されたの時のロールアップルールによる動作をテストします。デフォルト completed ロールアップルール「すべての子 Activity が completed なら親 Activity は completed になる」とデフォルト not satisfied ロールアップルール「すべての子 Activity が試行されるか not satisfied なら親 Activity は not satisfied になる」という 2 つのロールアップルールを用います。また動作は Retry Post Condition Rule を用いて確認します。

【構造】
Activity0 の Exit Rule を If completed, then exit
Activity0 の Post Condition Rule に If not satisfied, then retry
Activity1 の Limit Conditions の Attempt limit を 2
Activity1 の Precondition Rule を If attempt limit exceeded, then skip
Activity3 の Rollup Considerations を Required for Satisfied if not skipped
Activity3 の Rollup Rules の Rollup Progress Completion を false
Activity6 の Rollup Rules の Rollup Progress Completion を false
Activity6 の Rollup Rules の Rollup Objective Satisfied を false

【動作】
Step1 から 4 で Activity5 まで Continue リクエストで進みます。Activity1 の Satisfied Status を評価すると、Step3 で Activity3 の success_status に failed をセットしているため Activity1 は not satisfied となってます。

Step5 で Activity5 の success_status に passed を、progress_status に completed をセットします。Activity1、Activity6 の Rollup Progress Completion が false であるため、Activity0 の progress_status は Activity5 の progress_status で決まります。Activity5 が completed になったので Activity0 は completed になります。そのため Continue リクエストに対して、Exit Rule が成り立って Retry Post Condition Rule が評価されます。ここで Activity1 は not satisfied、Activity5 は satisfied、Activity6 は Rollup Rules の Rollup Objective Satisfied が false であることよりデフォルト not satisfied ロールアップルール評価対象とならず、Activity0 のデフォルト

not satisfied ロールアップルールが成り立ちます。従って、Activity0 は not satisfied となって Retry Post Condition Rule が成り立ち、Retry リクエストが発生して Activity2 が配信されます。

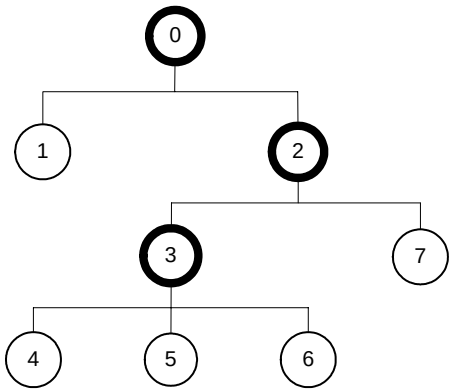
Step6 から 8 で Activity5 まで Continue リクエストで進みます。ここも前と同様に Activity1 の Satisfied Status を評価すると、Step6 で Activity2 の success_status に failed をセットしているため Activity1 は not satisfied となってます。

Step9 で Activity5 の success_status に failed を、progress_status に completed ををセットします。ここでも Step5 と同様に Activity0 のロールアップをみてみると、completed と not satisfied になり、Exit Rule が成り立って Retry Post Condition Rule が評価され、条件が成り立って Retry リクエストが発生します。今度も Activity0 を起点に Retry リクエストが発生していますが Activity1 の Limit Conditions の Attempt limit が 2 で Precondition Rule が If attempt limit exceeded, then skip のため Activity1 は Skip されて Activity5 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Flow に True をセットしています。

Test Case: RU-8a

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Post Condition Rule: If completed, then previous Rollup Rules: Completed if all Activity Progress Known and not completed
3	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If satisfied, then exit Post Condition Rule: If satisfied, then exit parent Rollup Rules: Incomplete if all Activity Progress Known or Objective Measure Known or Objective Status Known Satisfied if all attempted
4	Default
5	Default
6	Default
7	Rollup Considerations:

	Required for Incomplete if attempted

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 4 for delivery
3.	Set Activity 4's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
4.	Set Activity 5's cmi.score.scaled to `-0.5`; Process a *Continue* navigation request	Identify Activity 6 for delivery
5.	Set Activity 6's cmi.completion_status to `completed`; Process a *Continue* navigation request	Identify Activity 1 for delivery

【目的】
Rollup Considerations が設定されたの時のロールアップルールによる動作をテストします。
複数の Rollup Rules を使用し、Exit Parent Post Condition Rule を使用し確認します。

【構造】
Activity2 の Post Condition Rule を If satisfied, then exit
Activity2 の Rollup Rules を Completed if all Activity Progress Known and not completed
Activity3 の Exit Rule を If satisfied, then exit
Activity3 の Post Condition Rule を If satisfied, then exit parent
Activity3 の Rollup Rules を Incomplete if all Activity Progress Known or
Objective Measure Known or
Objective Status Known
Activity3 の Rollup Rules を Satisfied if all attempted
Activity7 の Rollup Considerations を Required for complete if attempted(*訳注)
(*訳注) 原本では Required for Incomplete if attempted となっていますが、対応する Testsuite の Manifest ファイルでは Required for complete if attempted となっています。

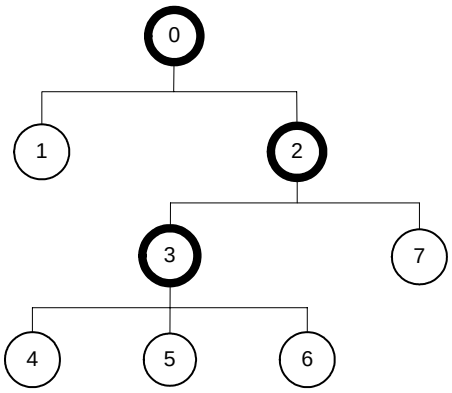
【動作】
Step1 から 5 で Activity6 まで以下のようにステータスをセットしながら Continue リクエストで進みます。
Step3 で Activity4 の success_status を failed、completion_status は自動的に completed
Step4 で Activity5 の score.scaled を-0.5、completion_status は自動的に completed
Step5 で Activity6 の completion_status を completed、success_status は自動的に passed

ここで Activity3、Activity2 のロールアップを実行すると、Activity3 の satisfied status は Rollup Rules より satisfed、progress status は Incomplete となります。また Activity7 が試行されていないため Activity2 の Completed ロールアップルールでは Activity3 だけが評価対象となって、Activity2 の progress status は Rollup Rules より completed となります。そのため Continue リクエストに対して Activity3 の Exit Rule が成り立って Exit parent Post Condition Rule が評価され、さらに Activity2 の Previous Post Condition Rule が評価され、条件が成り立って previous リクエストが発生し、Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: RU-8b

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Post Condition Rule: If completed, then previous Rollup Rules: Completed if all Activity Progress Known and not completed
3	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If satisfied, then exit Post Condition Rule: If satisfied, then exit parent Rollup Rules: Incomplete if all Activity Progress Known or Objective Measure Known or Objective Status Known Satisfied if all attempted
4	Delivery Controls: Objective Set by Content == true Completion Set by Content == true
5	Delivery Controls: Objective Set by Content == true

	Completion Set by Content == true
6	Delivery Controls: Objective Set by Content == true Completion Set by Content == true
7	Rollup Considerations: Required for Incomplete if attempted

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 4 for delivery
3.	Process a *Continue* navigation request	Identify Activity 5 for delivery
4.	Process a *Continue* navigation request	Identify Activity 6 for delivery
5.	Process a *Continue* navigation request	Identify Activity 7 for delivery

【目的】
Delivery Controls と Rollup Considerations が設定されたの時のロールアップルールによる動作をテストします。

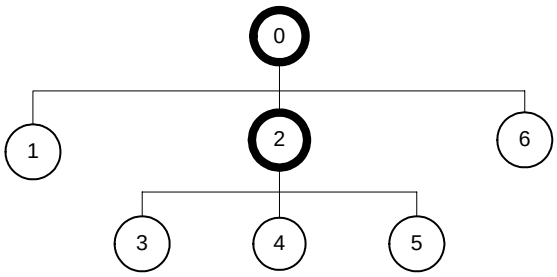
【構造】
Activity2 の Post Condition Rule を If satisfied, then exit
Activity2 の Rollup Rules を Completed if all Activity Progress Known and not completed
Activity3 の Exit Rule を If satisfied, then exit
Activity3 の Post Condition Rule を If satisfied, then exit parent
Activity3 の Rollup Rules を Incomplete if all Activity Progress Known or
Objective Measure Known or
Objective Status Known
Activity3 の Rollup Rules を Satisfied if all attempted
Activity4 の Delivery Controls の Objective Set by Content を true
Activity4 の Delivery Controls の Completion Set by Content を true
Activity5 の Delivery Controls の Objective Set by Content を true
Activity5 の Delivery Controls の Completion Set by Content を true
Activity6 の Delivery Controls の Objective Set by Content を true
Activity6 の Delivery Controls の Completion Set by Content を true
Activity7 の Rollup Considerations を Required for Incomplete if attempted

【動作】
Step1 から 5 で Activity6 まで Continue リクエストで進みます。Activity4,5,6 では Delivery Controls が設定されていて、SCO からなにもセットされないためステータスは変化ありません。そのためロールアップは発生せずに Step5 の Continue リクエストに対し Activity7 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています

Test Case: RU-9

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If satisfied, then exit Post Condition Rule: If not completed, then retry all If completed, then continue
3	Default
4	Limit Conditions: Attempt limit == 2 Sequencing Rules: Precondition Rule: If attempt limit exceeded, then skip Rollup Considerations: Required for Incomplete if not skipped Required for Completed if not skipped Required for Satisfied if not skipped
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.completion_status to `completed`;	Identify Activity 4 for delivery

	Process a *Continue* navigation request	
4.	Set Activity 4's cmi.completion_status to `incomplete`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.completion_status to `completed`; Process a *Continue* navigation request	Identify Activity 1 for delivery
6.	Process a *Continue* navigation request	Identify Activity 3 for delivery
7.	Set Activity 3's cmi.completion_status to `incomplete`; Process a *Continue* navigation request	Identify Activity 4 for delivery
8.	Set Activity 4's cmi.completion_status to `completed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
9.	Set Activity 5's cmi.completion_status to `completed`; Process a *Continue* navigation request	Identify Activity 1 for delivery
10.	Process a *Continue* navigation request	Identify Activity 3 for delivery
11.	Set Activity 3's cmi.completion_status to `completed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
12.	Set Activity 5's cmi.completion_status to `completed`; Process a *Previous* navigation request	Identify Activity 6 for delivery

【目的】
Limit Conditions と Rollup Considerations が設定されたの時のロールアップルールによる動作をテストします。デフォルト satisfied ロールアップルール「すべての子 Activity が satisfied なら親 Activity は satisfied になる」、デフォルト completed ロールアップルール「すべての子 Activity が completed なら親 Activity は completed になる」、デフォルト incomplete ロールアップルール「すべての子 Activity が試行されるか incomplete なら親 Activity は incomplete になる」という３つのロールアップルールを用います。動作は Retry All Post Condition Rule を用いて確認します。

【構造】
Activity2 の Exit Rule を If satisfied, then exit
Activity2 の Post Condition Rule を If not completed, then retry all
Activity2 の Post Condition Rule を If completed, then continue
Activity4 の Limit Conditions の Attempt limit を 2
Activity4 の Precondition Rule を If attempt limit exceeded, then skip
Activity4 の Rollup Considerations を Required for Incomplete if not skipped
Activity4 の Rollup Considerations を Required for Completed if not skipped
Activity4 の Rollup Considerations を Required for Satisfied if not skipped

【動作】
Step1 から 5 で Activity5 までステータスをセットしながら Continue リクエストで進みます。
Step3 で Activity3 の completion status に completed をセット、success status は自動的に passed
Step4 で Activity4 の completion status に incomplete をセット、success status は自動的に passed
Step5 で Activity5 の completion status に completed をセット、success status は自動的に passed
ここで Activity5 から別の Activity に移動しようとすると、Activity2 の子すべてが completed か incomplete のため、デフォルト incomplete ロールアップルールによって Activity2 は incomplete となります。また、Activity2 の子すべてが passed のため、デフォルト satisfied ロールアップルールによって Activity2 は satisfied となります。このため Exit Rule が成り立って Retry all Post Condition Rule が評価され、条件が成り立って Retry all リクエストが発生し

ます。このように Step5 の Continue リクエストの代わりに Retry all リクエストが実行されて Activity1 が配信されます。

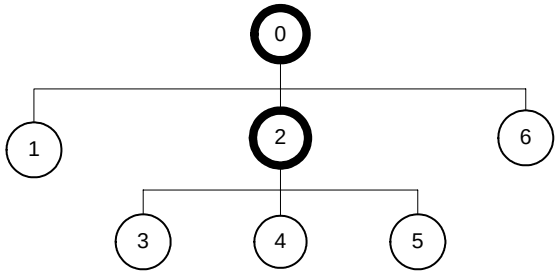
続いて前と同様に Step6 から 9 で Activity5 まで incomplete、completed、completed とステータスをセットしながら Continue リクエストで進みます。ここでも前と同様に Activity5 から別の Activity に移動しようとするとロールアップによって、Activity2 は satisfied かつ incomplete となり Exit Rule が成り立って Retry all Post Condition Rule が評価され、条件が成り立って Retry all リクエストが発生し Activity1 が配信されます。

再度前と同様に Step10 から 12 で Activity5 までステータスをセットしながら Continue リクエストで進みます。今度は Activity4 の試行が 3 度目になるので Precondition Rule が評価され Skip されます。Activity3 と Activity5 には completed、completed がセットされます。Activity4 には Required for Completed if not skipped がセットされているため、Activity3 と Activity5 が Activity2 のデフォルト complete ロールアップルールの評価対象となり Activity2 は satisfied かつ completed となります。Exit Rule が成り立って Continue Post Condition Rule が評価され、条件が成り立って Continue リクエストが発生します。このように Step5 の previous リクエストの代わりに Continue リクエストが実行されて Activity6 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています

Test Case: MS-1

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rule: Precondition Rule: If objective measure greater than 0.4, then skip
3	Rollup Rules: Rollup Objective Satisfied == false
4	Default
5	Default
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.score.scaled to `0.0`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 6 for delivery
6.	Process a *Previous* navigation request	Identify Activity 1 for delivery

【目的】
学習目標習得度のロールアップによる動作をテストします。動作は「学習目標習得度が 0.4 以上ならスキップする」という Precondition Rule を用いて確認します。

【構造】
Activity2 の Precondition Rule を If objective measure greater than 0.4, then skip

【動作】
Step3 から 5 で、Continue リクエストで Activity3 から 5 まで score.scaled に 1.0、0.0、1.0 とセットしながら進みます。

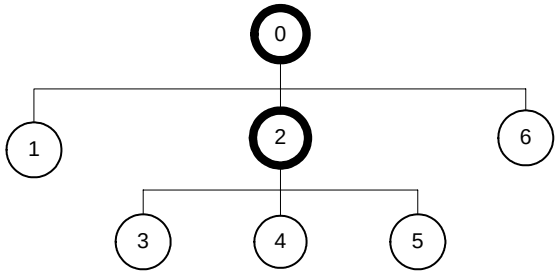
ここで Activity5 から別の Activity に移動しようとするときの習得度のロールアップで、Actibity2 の習得度が計算されます。Actibity3, 4, 5 の Rollup Objective Measure Weight はデフォルト値 1.0 なので、Actibity2 の習得度は (1.0×1.0 + 0.0×1.0 + 1.0×1.0)÷(1.0+1.0+1.0) = 2/3 となります。

次に Step6 の Previous リクエストで Activity2 の Precondition Rule の評価され、条件が成り立ち Activity2 は Skip されて Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。
Step3 の Rollup Objective Satisfied = false の設定も使われていません。

Test Case: MS-2

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rule: Precondition Rule: If objective measure less than 0.6, then skip
3	Rollup Rules: Rollup Objective Measure Weight == 0.75
4	Rollup Rules: Rollup Objective Measure Weight == 0.25
5	Rollup Rules: Rollup Objective Measure Weight == 0.25
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.score.scaled to `0.25`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.score.scaled to `0.5`; Process a *Continue* navigation request	Identify Activity 6 for delivery
6.	Process a *Previous* navigation request	Identify Activity 1 for delivery

【目的】
学習目標習得度のロールアップによる動作をテストします。動作は「学習目標習得度が 0.6 より小さかったらスキップする」という Precondition Rule を用いて確認します。

【構造】
Activity2 の Precondition Rule を If objective measure less than 0.6, then skip
Activity3 の Rollup Objective Measure Weight を 0.75
Activity4 の Rollup Objective Measure Weight を 0.25
Activity5 の Rollup Objective Measure Weight を 0.25

【動作】
Step3 から 5 で、Continue リクエストで Activity3 から 5 まで score.scaled に 0.25、1.0、0.5 とセットしながら進みます。

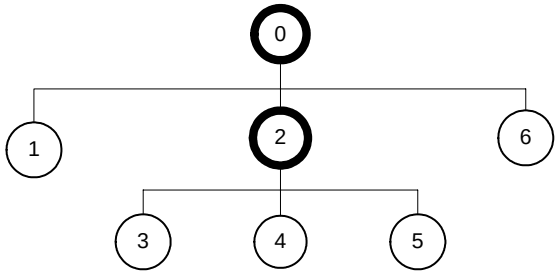
ここで Activity5 から別の Activity に移動しようとするときの習得度のロールアップで Actibity2 の習得度が計算されます。Activity3, 4, 5 の Rollup Objective Measure Weight を用いて Actibity2 の学習目標習得度を計算すると (0.25×0.75 + 1.0×0.25 + 0.5×0.25)÷ (0.75+0.25+0.25) = 0.45 となります。

次に Step6 の Previous リクエストで Activity2 の Precondition Rule の評価され、条件が成り立ち Activity2 は Skip されて Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: MS-3

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode; Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Precondition Rule: If objective measure greater than 0.50, then skip
3	Rollup Rules: Rollup Objective Measure Weight == 0.50
4	Rollup Rules: Rollup Objective Measure Weight == 0.00
5	Rollup Rules: Rollup Objective Measure Weight == 0.25
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.score.scaled to `-1.0`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.score.scaled to `0.5`; Process a *Continue* navigation request	Identify Activity 6 for delivery
6.	Process a *Previous* navigation request	Identify Activity 1 for delivery

【目的】
学習目標習得度のロールアップによる動作をテストします。動作は「学習目標習得度が 0.5 以上だったらスキップする」という Precondition Rule を用いて確認します。

【構造】
Activity2 の Precondition Rule を If objective measure greater than 0.50, then skip
Activity3 の Rollup Objective Measure Weight を 0.5
Activity4 の Rollup Objective Measure Weight を 0.0
Activity5 の Rollup Objective Measure Weight を 0.25

【動作】
Step3 から 5 で、Continue リクエストで Activity3 から 5 まで score.scaled に 1.0、-1.0、0.5 とセットしながら進みます。

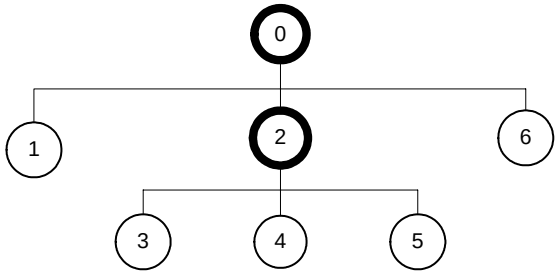
ここで Activity5 から別の Activity に移動しようとするときの習得度のロールアップで、Actibity2 の習得度が計算されます。Activity3, 4, 5 の Rollup Objective Measure Weight を用いて Actibity2 の学習目標習得度を計算すると (1.0×0.5 + -1.0×0.0 + 0.5×0.25)÷(0.5+0.0+0.25) = 0.816 となります。

次に Step6 の Previous リクエストで Activity2 の Precondition Rule の評価され、条件が成り立ち Activity2 は Skip されて Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: MS-4

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Precondition Rule: If objective measure less than 0.25, then skip
3	Default
4	Delivery Controls: Tracked = false
5	Rollup Rules: Rollup Objective Measure Weight == 0.25
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.score.scaled to `0.1`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.score.scaled to `0.5`; Process a *Continue* navigation request	Identify Activity 6 for delivery
6.	Process a *Previous* navigation request	Identify Activity 1 for delivery

【目的】
Delivery Controls の Tracked が false のケースを含む学習目標習得度のロールアップによる動作をテストします。動作は「学習目標習得度が 0.25 より小さいならスキップする」という Precondition Rule を用いて確認します。

【構造】
Activity2 の Precondition Rule を If objective measure less than 0.25, then skip
Activity4 の Delivery Controls の Tracked を false
Activity5 の Rollup Objective Measure Weight を 0.25

【動作】
Step3 から 5 で、Continue リクエストで Activity3 から 5 まで score.scaled に 0.1、1.0、0.5 とセットしながら進みます。

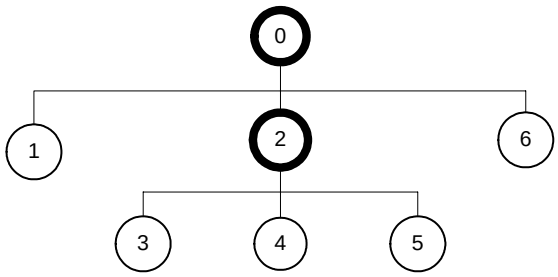
ここで Activity5 から別の Activity に移動しようとするときの習得度のロールアップで、Actibity2 の習得度が計算されます。Activity4 の Delivery Controls の Tracked が false であるため、Activity3, 5 の学習目標習得度と Rollup Objective Measure から Actibity2 の学習目標習得度を計算すると(0.1×1.0+0.5×0.25)÷(1.0+0.25) = 0.18 となります。

次に Step6 の Previous リクエストで Activity2 の Precondition Rule の評価され、条件が成り立ち Activity2 は Skip されて Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: MS-5a

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Objectives: Primary Objective: Objective Satisfied by Measure == true Minimum Normalized Measure == 0.6 Sequencing Rules: Exit Rule: If satisfied, then exit Post Condition Rule: If satisfied, then previous
3	Default
4	Default
5	Rollup Rules: Rollup Objective Measure Weight == 0.25
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Set Activity 4's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 1 for delivery

【目的】
Objective Satisfied by Measure で学習目標習得度からステータスが決定される場合のロールアップをテストします。動作は Previous Post Condition Rule を用いて確認します。

【構造】
Activity2 の Primary Objective の Objective Satisfied by Measure を true
Activity2 の Primary Objective の Minimum Normalized Measure を 0.6
Activity2 の Exit Rule を If satisfied, then exit
Activity2 の Post Condition Rule を If satisfied, then previous
Activity5 の Rollup Objective Measure Weight を 0.25

【動作】
Activity2 の学習目標習得度は、Activity3, 4, 5 の学習目標習得度と Rollup Objective Measure Weight から計算されます。学習目標習得度が設定されていない場合は 0 として計算されます。

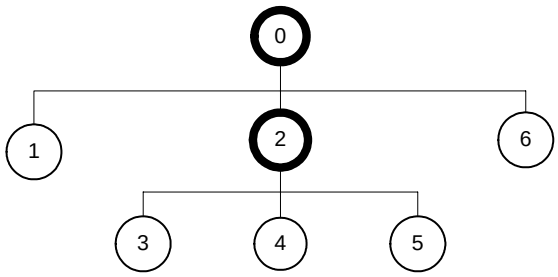
Step3 で Activity3 の SCO から score.scaled に 1.0 をセットしているので Activity2 の学習目標習得度は (1.0×1.0 + 0.0×1.0 + 0.0×0.25)÷(1.0+1.0+0.25) = 0.44 となります。Minimum Normalized Measure の値 0.6 を超えてないため Exit Rule の条件を満たさず Continue リクエストで Activity4 が配信されます。

Step4 で Activity4 の SCO から score.scaled に 1.0 セットしているので Activity2 の学習目標習得度は(1.0×1.0 + 1.0×1.0 + 0.0×0.25) (1.0+1.0+0.25) = 0.88 となります。Minimum Normalized Measure の値 0.6 を超えてますので Activity2 は satisfied になります。Exit Rule が成り立って Previous Post Condition Rule が評価され、条件がなりたって previous リクエストが発生します。そのため Step4 の Continue リクエストの代わりに previous リクエストが実行されて Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: MS-5b

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Objectives: Primary Objective: ObjectiveID == PRIMARYOBJ Objective Satisfied by Measure == true Minimum Normalized Measure == 0.6 Sequencing Rules: Exit Rule: If satisfied, then exit Post Condition Rule: If satisfied, then previous
3	Default
4	Default
5	Rollup Rules: Rollup Objective Measure Weight == 0.25
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.score.scaled to `0.5`; Set Activity 3's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 4 for delivery

4.	Set Activity 4's cmi.score.scaled to `0.75`; Set Activity 4's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.score.scaled to `1.0`; Set Activity 5's cmi.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 1 for delivery

【目的】
Objective Satisfied by Measure で学習目標習得度からステータスが決定される場合のロールアップをテストします。動作は Previous Post Condition Rule を用いて確認します。

【構造】
Activity2 の Primary Objective の Objective Satisfied by Measure を true
Activity2 の Primary Objective の Minimum Normalized Measure を 0.6
Activity2 の Exit Rule を If satisfied, then exit
Activity2 の Post Condition Rule を If satisfied, then previous
Activity5 の Rollup Objective Measure Weight を 0.25

【動作】
Step3 で Activity3 の SCO から score.scaled に 0.5 セットしているので Activity2 の学習目標習得度は (0.5×1.0 + 0.0×1.0 + 0.0×0.25)÷(1.0+1.0+0.25) = 0.22 となります。Minimum Normalized Measure の値 0.6 を超えてないため Exit Rule の条件を満たさず Continue リクエストで Activity4 が配信されます。

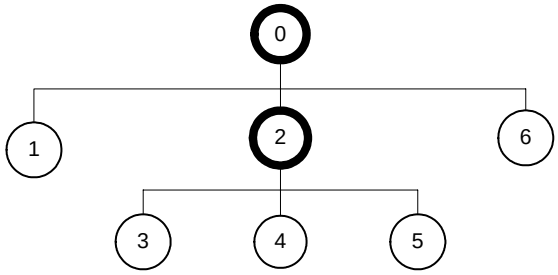
Step4 で Activity4 の SCO から score.scaled に 0.75 セットしているので Activity2 の学習目標習得度は(0.5×1.0 + 0.75×1.0 + 0.0×0.25)÷(1.0+1.0+0.25) = 0.55 となります。ここでもまだ Minimum Normalized Measure の値 0.6 を超えてないため Exit Rule の条件を満たさず Continue リクエストで Activity5 が配信されます。

Step4 で Activity4 の SCO から score.scaled に 1.0 セットしているので Activity2 の学習目標習得度は(0.5×1.0 + 0.75×1.0 + 1.0×0.25)÷(1.0+1.0+0.25)= 0.66 となります。Minimum Normalized Measure の値 0.6 を超えてますので Activity2 は satisfied になります。Exit Rule が成り立って Previous Post Condition Rule が評価され、条件がなりたって previous リクエストが発生します。そのため Step4 の Continue リクエストの代わりに previous リクエストが実行されて Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。
Activity2 の子 Activity のすべてで success_status に failed をセットしていますが、ここでは習得度で status を判断しているため使われていません。

Test Case: MS-6

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Objectives: Primary Objective: Objective Satisfied by Measure == true Minimum Normalized Measure == 0.6 Sequencing Rules: Exit Rule: If satisfied, then exit Post Condition Rule: If satisfied, then previous Precondition Rule: If satisfied, then skip Rollup Considerations: Measure Satisfaction If Active == false
3	Default
4	Default
5	Rollup Rules: Rollup Objective Measure Weight == 0.25
6	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.score.scaled to `1.0`;	Identify Activity 4 for delivery

	Process a *Continue* navigation request	
4.	Set Activity 4's cmi.score.scaled to `0.75`; Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.score.scaled to `0.5`; Process a *Continue* navigation request	Identify Activity 6 for delivery
6.	Process a *Previous* navigation request	Identify Activity 1 for delivery

【目的】Objective Satisfied by Measure が true，Measure Satisfaction If Active が false で，学習目標習得度からステータスが決定される場合のロールアップをテストします。動作は Previous Precondition Rule を用いて確認します。

【構造】
Activity2 の Primary Objective の Objective Satisfied by Measure を true
Activity2 の Primary Objective の Minimum Normalized Measure を 0.6
Activity2 の Exit Rule を If satisfied, then exit
Activity2 の Post Condition Rule を If satisfied, then previous
Activity2 の Precondition Rule を If satisfied, then skip
Activity2 の Rollup Considerations の Measure Satisfaction If Active を false
Activity5 の Rollup Objective Measure Weight を 0.25

【動作】
Step3 から 5 で、Activity3 から 5 まで score.scaled に 1.0、0.75、0.5 とセットしていますので、Actibity2 の学習目標習得度は、Activity3 から 5 の Rollup Objective Measure Weight を用いて計算されます。ただし、Activity2 の Measure Satisfaction If Active が false のため、Step3 から 5 で Activity2 が Active の間は、学習目標習得度と Minimum Normalized Measure の比較による学習目標のロールアップは起きません。

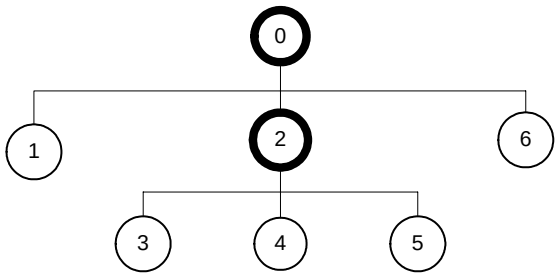
Step5 の段階で、Activity2 の学習目標習得度は (1.0×1.0 + 0.75×1.0 + 0.5×0.25)÷ (1.0+1.0+0.25) = 0.83 となり Minimum Normalized Measure の値 0.6 を超えています。また、このとき Continue リクエストによって Activity2 が終了するので、学習目標習得度と Minimum Normalized Measure の比較が行われ Activity2 は satisfied になります。

次に Step6 の Previous リクエストで Activity2 の Precondition Rule の評価され、条件が成り立ち Activity2 は Skip されて Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: SX-2

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Default
2	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If satisfied, then exit Post Condition Rule: If satisfied, then previous Rollup Rules: Satisfied if all completed
3	Completion Threshold** == 0.5
4	Default
5	Completion Threshold** == 0.75
6	Default

** This element is in the adlcp namespace.

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Set Activity 3's cmi.completion_status to `incomplete`; Set Activity 3's cmi.progress_measure to `0.5`; Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.progress_measure to `0.9`;	Identify Activity 1 for delivery

	Process a *Continue* navigation request	

【目的】
Completion Threshold のテストを行います。「すべての子 Activity が Completed ならば親 Activity は Satisfied とする」という Rollup Rules と、Previous Post Condition Rule を用いて動作を確認します。

【構造】
Activity2 の Exit Rule を If satisfied, then exit
Activity2 の Post Condition Rule を If satisfied, then previous
Activity2 の Rollup Rules を Satisfied if all completed
Activity3 の Completion Threshold を 0.5
Activity5 の Completion Threshold を 0.5

【動作】
Step3 で Activity3 の SCO から completion_status に incomplete がセットされていますが、progress_measure に 0.5 がセットされているため、Completion Threshold の比較で completed になります。しかし 3 個ある子 Activity のうちの 1 つの子 Activity の completion_status が completed になっただけなので、Activity2 の Rollup Rules は成り立たず Continue リクエストで Activity4 が配信されます。

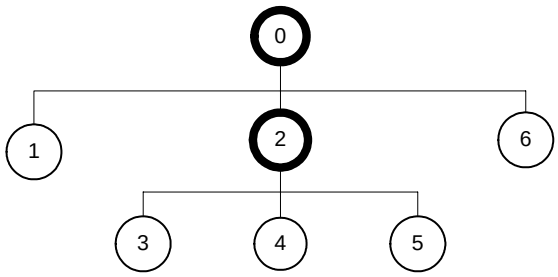
Step4 では Activity4 の SCO からなにもセットされないため、自動的に completed になります。しかしまだ 3 個ある子 Activity のうちの 2 つの子 Activity の completion_status が completed になっただけなので、Activity2 の Rollup Rules は成り立たず Continue リクエストで Activity5 が配信されます。

Step5 では Activity5 の SCO から progress_measure に 0.9 がセットされているため Completion Threshold の比較で completed になります。これによって Activity2 の子 Activity すべてが completed になるため Rollup Rule が成り立って Activity2 が Completed になります。そのため、Continue リクエストに対して、Exit Rule が成り立って Previous Post Condition Rule が評価され、条件が成り立って previous リクエストが発生し、Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: SX-3

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Objectives: Primary Objective: *empty* Objective: Objective ID == obj1 Map Info: Target Objective ID == gObj-SX3-1 Read Satisfied Status == false Read Normalized Measure == false Write Satisfied Status == true Write Normalized Measure == true Objective: Objective ID == obj2 Map Info: Target Objective ID == gObj-SX3-2 Read Satisfied Status == false Read Normalized Measure == false Write Satisfied Status == true Write Normalized Measure == true Objective: Objective ID == obj3 Map Info: Target Objective ID == gObj-SX3-3 Read Satisfied Status == false Read Normalized Measure == false Write Satisfied Status == true Write Normalized Measure == true
2 **	Control Mode: Flow == true Choice == false Sequencing Rules:

	Exit Rule: If completed, then exit Post Condition Rule: If completed, then previous Rollup Rules: Completed if all satisfied
3	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Objective Satisfied by Measure == true Minimum Normalized Measure == 0.6 Map Info: Target Objective ID == gObj-SX3-1
4	Default
5	Default
6	Default

****** Subtree 2 is represented as a (sub)manifest in the content package.

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Find the index of Activity 1's cmi.objectives.n with ID of `obj1`; Set that objective's score.scaled to `0.75`; Set that objective's cmi.objectives.n.success_status to `failed`; Find the index of Activity 1's cmi.objectives.n with ID of `obj2`; Set that objective's cmi.objectives.n.score.scaled to `0.90`; Set that objective's cmi.objectives.n.success_status to `passed`; Find the index of Activity 1's cmi.objectives.n with ID of `obj3`; Set that objective's cmi.objectives.n.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 3 for delivery
3.	Process a *Continue* navigation request	Identify Activity 4 for delivery
4.	Process a *Continue* navigation request	Identify Activity 5 for delivery
5.	Set Activity 5's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 1 for delivery

【目的】
Sub Manifest のテストを行います。本体の Manifest と Sub Manifest で共有される学習目標を定義し、値の変化が反映されることを確認します。確認は「すべての子 Activity が Completed ならば親 Activity は Satisfied とする」という Rollup Rules と Previous Post ConditionRule を用いて行います。

【構造】
Activity1 で共有学習目標 gObj-SX3-1 から gObj-SX3-3 を定義。学習目標習得状態、学習目標習得度は共有学習目標へ書き込み
Activity2 の Exit Rule を If completed, then exit
Activity2 の Post Condition Rule を If completed, then previous
Activity2 の Rollup Rules を Completed if all satisfied
Activity3 で共有学習目標 gObj-SX3-1 を定義。学習目標習得状態、学習目標習得度はデフォルトで共有学習目標から読み込み
Activity3 の Objective Satisfied by Measure を true
Activity3 の Minimum Normalized Measure を 0.6

【動作】
Step2 で Activity1 から共有学習目標を設定します。obj1 は gObj-SX3-1 に書き込むようにマップされていますので、gObj-SX3-1 の学習目標習得状態は failed、学習目標習得度は 0.75 になります。これによって、gObj-SX3-1 を経由して Activity3 へのロールアップが発生します。Activity3 では Objective Satisfied by Measure を true なので、Minimum Normalized Measure＝0.6 と学習目標習得度＝0.75 の比較が行われ、Activity3 は satisfied になります。

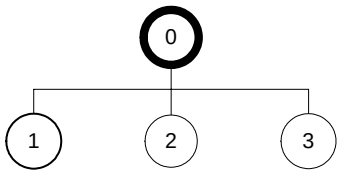
Step3, 4, 5 で Activity3, 4, 5 と異動します。各 Activity のステータスは以下のようになります。
Activity3 は、先ほどのロールアップで satisfied
Activity4 は Step4 で SCO からなにもセットされないため自動的に satisfied
Activity5 は Step5 で success status に passed をセット

以上で Activity2 の子すべてが satisfied のため Rollup Rule が成り立ち Activity2 は completed となります。そのため Continue リクエストに対して、Exit Rule が成り立って Previous Post Condition Rule が評価され、条件が成り立って previous リクエストが発生し、Activity1 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: OB-1a

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Objectives: Primary Objective: ID == PRIMARYOBJ_1 Map Info: Target Objective ID == gObj-OB1a Write Satisfied Status == true
2	Sequencing Rules: Precondition Rule: If satisfied, then skip Objectives: Primary Objective: ID == PRIMARYOBJ_2 Map Info: Target Objective ID == gObj-OB1a
3	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Set Activity 1's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 3 for delivery

【目的】
ここでは共有学習目標の Satisfied Status の参照をテストします。結果は Precondition Rule を用いて確認します。

【構造】
共有学習目標 gObj-OB1a を使用する。

Activity1 の Primary Objective の ID を PRIMARYOBJ_1
Activity1 の Primary Objective を gObj-OB1a にマップ
Activity1 の Primary Objective のマップ情報として Write Satisfied Status を true
Activity2 の Precondition Rule を If satisfied, then skip
Activity2 の Primary Objective の ID を PRIMARYOBJ_2
Activity2 の Primary Objective を gObj-OB1a にマップ

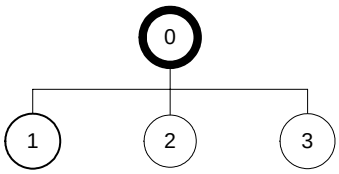
【動作】
Step2 で SCO から success_status に passed をセットすることで Activity1 は satisfied となり、ここにマップされている共有学習目標 gObj-OB1a も Write Satisfied Status を true であることから satisfied となります。

Continue リクエストで配信可能な Activity を探します。Activity2 では、マップされている共有学習目標 gObj-OB1a が satisfied であることから、Precondition Rule の条件を満たすため Skip されて Activity3 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: OB-1b

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Objectives: Primary Objective: Map Info: Target Objective ID == gObj-OB1b Write Normalized Measure == true
2	Sequencing Rules: Precondition Rule: If objective measure greater than 0.75, then skip Objectives: Primary Objective: ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-OB1b
3	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Set Activity 1's cmi.score.scaled to `0.8`; Process a *Continue* navigation request	Identify Activity 3 for delivery

【目的】
共有学習目標の習得度の参照をテストします。結果は Precondition Rule を用いて確認します。

【構造】
共有学習目標 gObj-OB1b を使用する。
Activity1 の Primary Objective を gObj-OB1b にマップ
Activity1 の Primary Objective のマップ情報として Write Normalized Measure を true

Activity2 の Precondition Rule を If objective measure greater than 0.75, then skip
Activity2 の Primary Objective の ID を PRIMARYOBJ
Activity2 の Primary Objective を gObj-OB1b にマップ

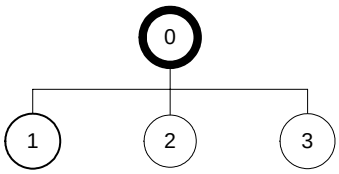
【動作】
Step2 で SCO から score.scaled に 0.8 をセットすることで Activity1 の習得度は 0.8 となり、ここにマップされている共有学習目標 gObj-OB1b の習得度も Write Normalized Measure を true より 0.8 となります。

Continue リクエストで配信可能な Activity を探します。Activity2 では、マップされている共有学習目標 gObj-OB1b の習得度が 0.8 であることから、Precondition Rule の条件を満たすため Skip されて Activity3 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: OB-1b

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-OB1c Write Satisfied Status == true Write Normalized Measure == true
2	Sequencing Rules: Precondition Rule: If satisfied, then skip Objectives: Primary Objective: Objective ID == PRIMARYOBJ Objective Satisfied by Measure == true; Minimum Normalized Measure == 0.50 Map Info: Target Objective ID == gObj-OB1c
3	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Set Activity 1's cmi.success_status to `failed`; Set Activity 1's cmi.score.scaled to `0.8`; Process a *Continue* navigation request	Identify Activity 3 for delivery

【目的】
ここでは共有学習目標の習得度と satisfied status の参照をテストします。結果は Precondition Rule を用いて確認します。

【構造】
共有学習目標 gObj-OB1c を使用する。
Activity1 の Primary Objective の ID を PRIMARYOBJ
Activity1 の Primary Objective を gObj-OB1c にマップ
Activity1 の Primary Objective のマップ情報として Write Satisfied Status を true
Activity1 の Primary Objective のマップ情報として Write Normalized Measure を true
Activity2 の Precondition Rule を If satisfied, then skip
Activity2 の Primary Objective の ID を PRIMARYOBJ
Activity2 の Primary Objective の Objective Satisfied by Measure を true
Activity2 の Primary Objective の Write Normalized Measure を true
Activity2 の Primary Objective を gObj-OB1b にマップ

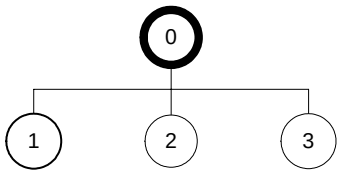
【動作】
Step2 で SCO から success_status に failed、score.scaled に 0.8 をセットすることで Activity1 のステータスは failed で習得度は 0.8 となります。またマップされている共有学習目標 gObj-OB1c も Write Satisfied Status が true 、Write Normalized Measure が true であることからステータスは not satisfied、習得度は 0.8 となります。

Continue リクエストで配信可能な Activity を探します。Activity2 ではマップされている共有学習目標 gObj-OB1c からステータスを取得します。gObj-OB1c は failed になっていますが、ここでは Objective Satisfied by Measure が true のため、こちらが優先されるため、習得度 0.8 との比較が評価され、Precondition Rule の条件を満たすため Skip されて Activity3 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています

Test Case: OB-2a

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Objectives: Primary Objective: *empty* Objective: Objective ID == obj1 Map Info: Target Objective ID == gObj-OB2a Write Satisfied Status == true
2	Sequencing Rules: Precondition Rule: If obj1 not satisfied, then skip Objectives: Primary Objective: *empty* Objective: Objective ID == obj1 Map Info: Target Objective ID == gObj-OB2a
3	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Set Activity 1's cmi.success_status to `passed`; Find the index of Activity 1's cmi.objectives.n with ID of `obj1`; Set that objective's cmi.objectives.n.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 3 for delivery

【目的】

Promary Objective ではない共有学習目標の satisfied status の参照をテストします。結果は「指

定した local object が not satisfied であれば Skip する」Precondition Rule を用いて確認します。

【構造】
共有学習目標 gObj-OB2a を使用する。
Activity1 の Primary Objective を empty
Activity1 に ID:obj1 の Local Objective を作成
Activity1 の ID:obj1 の Local Objective を gObj-OB2a にマップ
Activity1 の ID:obj1 の Local Objective のマップ情報として Write Satisfied Status を true
Activity2 の Precondition Rule を If obj1 not satisfied, then skip
Activity2 の Primary Objective を empty
Activity2 に ID:obj1 の Local Objective を作成
Activity2 の ID:obj1 の Local Objective を gObj-OB2a にマップ

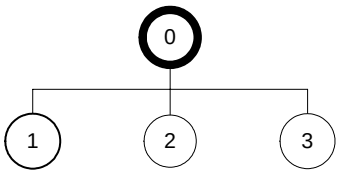
【動作】
Step2 で SCO から ID が obj1 の objective を探し、success_status に failed をセットしています。
Activity1 の IDobj1 がマップされている共有学習目標 gObj-OB2a の Write Satisfied Status が true であることから、gObj-OB2a は not satisfied になります。

Continue リクエストで配信可能な Activity を探します。Activity2 では、ID が obj1 の local objective にマップされている共有学習目標 gObj-OB2a からステータスを取得し not satisfied であることから Precondition Rule が成り立ち Skip されて Activity3 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。
また SCO から success_status がセットされていますが、こちらは使われていません。

Test Case: OB-2b

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Objectives: Primary Objective: *empty* Objective: Objective ID == obj1 Map Info: Target Objective ID == gObj-OB2b Write Normalized Measure == true
2	Sequencing Rule: Precondition Rule: If obj1 objective measure less than 0.25, then skip Objectives: Primary Objective: *empty* Objective: Objective ID == obj1 Map Info: Target Objective ID == gObj-OB2b
3	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Find the index of Activity 1's cmi.objectives.n with ID of `obj1`; Set that objective's cmi.objectives.n.score.scaled to `0.0`; Process a *Continue* navigation request	Identify Activity 3 for delivery

【目的】
Promary Objective ではない共有学習目標の習得度の参照をテストします。結果は「指定した local object の習得度が 0.25 より小さければ Skip する」Precondition Rule を用いて確認します。

【構造】
共有学習目標 gObj-OB2b を使用する。
Activity1 の Primary Objective を empty
Activity1 に ID:obj1 の Local Objective を作成
Activity1 の ID:obj1 の Local Objective を gObj-OB2a にマップ
Activity1 の ID:obj1 の Local Objective のマップ情報として Write Normalized Measure を true
Activity2 の Precondition Rule を If obj1 objective measure less than 0.25, then skip
Activity2 の Primary Objective を empty
Activity2 に ID:obj1 の Local Objective を作成
Activity2 の ID:obj1 の Local Objective を gObj-OB2b にマップ

【動作】
Step2 では SCO から ID が obj1 の objective を探し、score.scaled に 0.0 をセットしているため、Activity1 の IDobj1 がマップされている共有学習目標 gObj-OB2b の Write Normalized Measure が true であることから、gObj-OB2a の習得度は 0.0 になります。

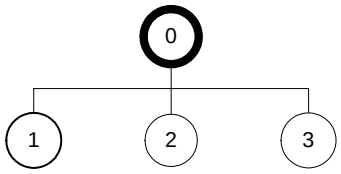
Continue リクエストで配信可能な Activity を探します。Activity2 では、ID が obj1 の local objective にマップされている共有学習目標 gObj-OB2b から習得度を取得し、0.0 であることから Precondition Rule の条件が成り立ち Skip されて Activity3 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: OB-3a

NOTE: Test Cases OB-3a and OB-3b and OB-3c test cross-activity tree persistence of global shared objectives; they should be performed back to back – OB-3a first, followed by OB-3b, and followed by OB-3 c.

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Objectives: Primary Objective: *empty* Objective: Objective ID == obj1 Map Info: Target Objective ID == gObj-OB3-1 Write Normalized Measure == true Write Satisfied Status == true
2	Objectives: Primary Objective: *empty* Objective: Objective ID == obj1 Map Info: Target Objective ID == gObj-OB3-2 Write Normalized Measure == true Write Satisfied Status == true
3	Objectives: Primary Objective: *empty* Objective: Objective ID == obj1 Map Info: Target Objective ID == gObj-OB3-3 Write Normalized Measure == true Write Satisfied Status == true

Test Script:

Step	Action	Expected Result

1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Find the index of Activity 1's cmi.objectives.n with ID of `obj1`; Set that objective's cmi.objectives.n.score.scaled to `0.0`; Set that objective's cmi.objectives.n.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 2 for delivery
3.	Find the index of Activity 2's cmi.objectives.n with ID of `obj1`; Set that objective's cmi.objectives.n.score.scaled to `0.5`; Set that objective's cmi.objectives.n.success_status to `failed`; Process a *Continue* navigation request	Identify Activity 3 for delivery
4.	Find the index of Activity 3's cmi.objectives.n with ID of `obj1`; Set that objective's cmi.objectives.n.score.scaled to `0.75`; Process a *Exit All* navigation request	End Sequencing Session

【目的】
コンテンツをまたがる共有学習目標のテストします。ここでは共有学習目標にセットするだけで特にシーケンシングのテストはおこないません。コンテンツ OB-3b と OB-3c で動作を確認します。

【構造】
共有学習目標 gObj-OB3-1、gObj-OB3-2、gObj-OB3-3 を使用する。
Activity1 の Primary Objective を empty
Activity1 に ID:obj1 の Local Objective を作成
Activity1 の ID:obj1 の Local Objective を gObj-OB3-1 にマップ
Activity1 の ID:obj1 の Local Objective のマップ情報として Write Normalized Measure を true
Activity1 の ID:obj1 の Local Objective のマップ情報として Write Satisfied Status を true
Activity2 の Primary Objective を empty
Activity2 の ID:obj1 の Local Objective を gObj-OB3-2 にマップ
Activity2 の ID:obj1 の Local Objective のマップ情報として Write Normalized Measure を true
Activity2 の ID:obj1 の Local Objective のマップ情報として Write Satisfied Status を true
Activity3 の Primary Objective を empty
Activity3 の ID:obj1 の Local Objective を gObj-OB3-3 にマップ
Activity3 の ID:obj1 の Local Objective のマップ情報として Write Normalized Measure を true
Activity3 の ID:obj1 の Local Objective のマップ情報として Write Satisfied Status を true

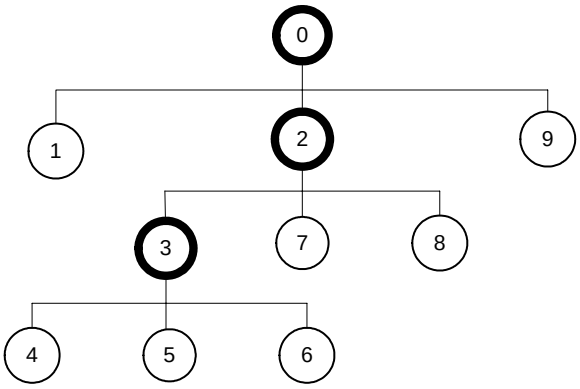
【動作】
Step2～5 で Activity0 の子 Activity に success_status と score.scaled を以下のようにセットしながら進みます。
Activity1 の Step2 で success_status に passed を、score.scaled に 0.0 をセット
Activity2 の Step3 で success_status に faild を、score.scaled に 0.5 をセット
Activity3 の Step4 で、score.scaled に 0.0 をセット
以上で終了です。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: OB-3b

NOTE: Test Cases OB-3a and OB-3b and OB-3c test cross-activity tree persistence of global shared objectives; they should be performed backed to back – OB-3a first, followed by OB-3b, and followed by OB-3 c.

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0 **	Control Mode: Flow == true Choice == false
1	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-OB3-2 Sequencing Rules: Precondition Rule: If not satisfied, then skip
2	Control Mode: Flow == true Choice == false
3	Control Mode: Flow == true Choice == false Objectives: Primary Objective: Objective ID == PRIMARYOBJ Objective Satisfied by Measure == true Minimum Normalized Measure == 0.6 Map Info: Target Objective ID == gObj-OB3-3 Read Satisfied Status == false

	Write Satisfied Status == true Sequencing Rules: Precondition Rule: If satisfied, then skip
4	Default
5	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-OB3-1 Sequencing Rules: Precondition Rule: If satisfied, then skip
6	Default
7	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-OB3-3 Sequencing Rules: Precondition Rule: If satisfied, then skip
8	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-OB3-3 Sequencing Rules: Precondition Rule: If satisfied, then skip
9	Default

** This activity (the root of the activity tree) has Objectives Global to System == false.

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 4 for delivery
3.	Set Activity 4's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 5 for delivery
4.	Set Activity 5's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 6 for delivery
5.	Set Activity 6's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 9 for delivery

【目的】
コンテンツをまたがる共有学習目標のテストします。Objectives Global to System が false の場合です。

【構造】
共有学習目標 gObj-OB3-1、gObj-OB3-2、gObj-OB3-3 を使用する。
Activity1 の Primary Objective の Objective ID を PRIMARYOBJ
Activity1 の Primary Objective を gObj-OB3-2 にマップ
Activity1 の Precondition Rule を If not satisfied, then skip
Activity3 の Primary Objective の Objective ID を PRIMARYOBJ
Activity3 の Primary Objective の Objective Satisfied by Measure を true
Activity3 の Primary Objective の Minimum Normalized Measure を 0.6
Activity3 の Primary Objective を gObj-OB3-3 にマップ
Activity3 の Primary Objective のマップ情報として Read Satisfied Status を false
Activity3 の Primary Objective のマップ情報として Write Satisfied Status を true
Activity3 の Precondition Rule を If satisfied, then skip
Activity5 の Primary Objective の Objective ID を PRIMARYOBJ
Activity5 の Primary Objective を gObj-OB3-1 にマップ
Activity5 の Precondition Rule を If satisfied, then skip
Activity7 の Primary Objective の Objective ID を PRIMARYOBJ
Activity7 の Primary Objective を gObj-OB3-3 にマップ
Activity7 の Precondition Rule を If satisfied, then skip
Activity8 の Primary Objective の Objective ID を PRIMARYOBJ
Activity8 の Primary Objective を gObj-OB3-3 にマップ
Activity8 の Precondition Rule を If satisfied, then skip

【動作】
Objectives Global to System が false のため OB-3a で共有学習目標にセットしたデータは使用しません。

Step3 から 5 で Activity3 から 5 まで score.scaled に 1.0、1.0、1.0 とセットしながら Continue リクエストで Activity6 まで進みます。

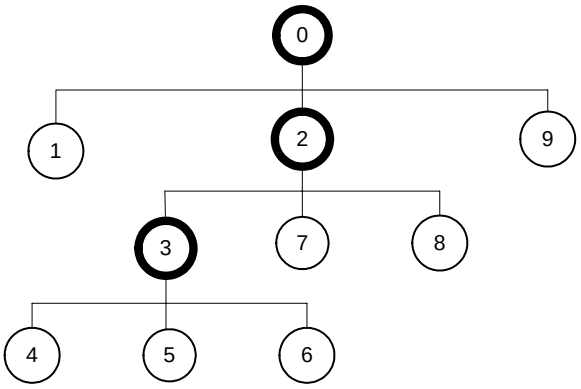
Step5 の Continue リクエストで Activity3 の習得度ロールアップが発生し Activity3 の習得度を計算すると、(1.0+1.0+1.0) ÷(1.0+1.0+1.0)=1.0 になり、Minimum Normalized Measure と比較すし Satisfied status は satisfied となります。また Activity3 は gObj-OB3-3 にマップされており、Write Satisfied Status が true であることから gObj-OB3-3 の Satisfied status も satisfied となります。Continue リクエストで配信可能な Activity と探すと Activity7 はマップされている gObj-OB3-3 の Satisfied status が satisfied のため Precondition Rule を満たすためスキップ、同様に Activity8 もスキップして Activity 9 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: OB-3c

NOTE: Test Cases OB-3a and OB-3b and OB-3c test cross-activity tree persistence of global shared objectives; they should be performed backed to back – OB-3a first, followed by OB-3b, and followed by OB-3 c.

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-OB3-3 Sequencing Rules: Precondition Rule: If objective status not known, then skip
2	Control Mode: Flow == true Choice == false
3	Control Mode: Flow == true Choice == false Objectives: Primary Objective: Objective ID == PRIMARYOBJ Objective Satisfied by Measure == true Minimum Normalized Measure == 0.6 Map Info: Target Objective ID == gObj-OB3-3 Read Satisfied Status == false

	Write Satisfied Status == true Sequencing Rules: Precondition Rule: If satisfied, then skip
4	Default
5	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-OB3-1 Sequencing Rules: Precondition Rule: If satisfied, then skip
6	Default
7	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-OB3-2 Sequencing Rules: Precondition Rule: If not satisfied, then skip
8	Objectives: Primary Objective: Map Info: Target Objective ID == gObj-OB3-2 Sequencing Rules: Precondition Rule: If not satisfied, then skip
9	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 9 for delivery

【目的】
コンテンツをまたがる共有学習目標のテストします。

【構造】
共有学習目標 gObj-OB3-1、gObj-OB3-2、gObj-OB3-3 を使用する。
Activity1 の Primary Objective の Objective ID を PRIMARYOBJ
Activity1 の Primary Objective を gObj-OB3-3 にマップ
Activity1 の Precondition Rule を If objective status not known, then skip
Activity3 の Primary Objective の Objective ID を PRIMARYOBJ
Activity3 の Primary Objective の Objective Satisfied by Measure を true
Activity3 の Primary Objective の Minimum Normalized Measure を 0.6
Activity3 の Primary Objective を gObj-OB3-3 にマップ
Activity3 の Primary Objective のマップ情報として Read Satisfied Status を false
Activity3 の Primary Objective のマップ情報として Write Satisfied Status を true

Activity3 の Precondition Rule を If satisfied, then skip
Activity5 の Primary Objective の Objective ID を PRIMARYOBJ
Activity5 の Primary Objective を gObj-OB3-1 にマップ
Activity5 の Precondition Rule を If satisfied, then skip
Activity7 の Primary Objective の Objective ID を PRIMARYOBJ
Activity7 の Primary Objective を gObj-OB3-2 にマップ
Activity7 の Precondition Rule を If not satisfied, then skip
Activity8 の Primary Objective の Objective ID を PRIMARYOBJ
Activity8 の Primary Objective を gObj-OB3-2 にマップ
Activity8 の Precondition Rule を If satisfied, then skip

【動作】
OB-3a で共有学習目標にセットしたデータを使用します。OB-3a でセットされた情報は
gObj-OB3-1 は satisfied status が satisfied、習得度は 0.0
gObj-OB3-2 は satisfied status が not satisfied、習得度は 0.5
gObj-OB3-3 は satisfied status が unknown、習得度は 0.75

Start リクエストで配信 Activity を探すと以下のようになります。

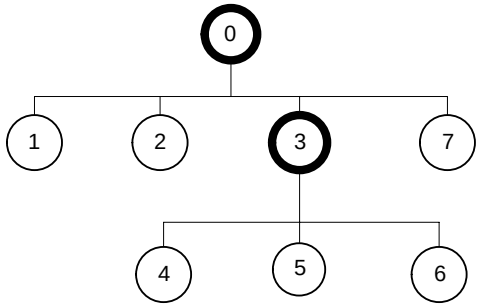
Activity1 はマップされている gObj-OB3-3 の satisfied status が unknown のため Skip
Activity3 は マップされている gObj-OB3-3 の習得度が 0.75、Minimum
Normalized Measure が 0.6 のため satisfied となり Precondition Rule の条件を満たし Skip
Activity7 はマップされている gObj-OB3-2 の satisfied status が not satisfied のため Skip
Activity8 はマップされている gObj-OB3-2 の satisfied status が not satisfied のため Skip

この結果 Activity9 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: OB-4

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-OB4-1 Read Satisfied Status == false Read Normalized Measure == false Write Satisfied Status == true Map Info: Target Objective ID == gObj-OB4-3 Read Satisfied Status == false Read Normalized Measure == false Write Normalized Measure == true
2	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-OB4-2 Write Satisfied Status == true
3	Control Mode: Flow == true Choice == false Sequencing Rules: Precondition Rule: If completed, then skip Rollup Rules: Completed if all Satisfied

4	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-OB4-1
5	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-OB4-2
6	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Objective Satisfied by Measure == true Minimum Normalized Measure == -0.75 Map Info: Target Objective ID == gObj-OB4-3
7	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Set Activity 1's cmi.score.scaled to `-0.25`; Process a *Continue* navigation request	Identify Activity 2 for delivery
3.	Set Activity 2's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 7 for delivery

【目的】
共有学習目標を使用し、「すべての子 Activity が satisfied なら親 Activity は completed になる」という Rollup Rules による動作をテストします。結果は「completed なら Skip する」Precondition Rule をもちいて確認します。

【構造】
共有学習目標として gObj-OB4-1、gObj-OB4-2、gObj-OB4-3 を使用
Activity1 の Primary Objective の ID を PRIMARYOBJ
Activity1 に Primary Objective のマップ情報 1 として Target Objective ID を gObj-OB4-1
Activity1 の Primary Objective のマップ情報 1 として Read Satisfied Status を false
Activity1 の Primary Objective のマップ情報 1 として Read Normalized Measure を false
Activity1 の Primary Objective のマップ情報 1 として Write Satisfied Status を true
Activity1 に Primary Objective のマップ情報 2 として Target Objective ID を gObj-OB4-3
Activity1 の Primary Objective のマップ情報 2 として Read Satisfied Status を false
Activity1 の Primary Objective のマップ情報 2 として Read Normalized Measure を false
Activity1 の Primary Objective のマップ情報 2 として Write Normalized Measure を true
Activity2 の Primary Objective の ID を PRIMARYOBJ
Activity2 に Primary Objective のマップ情報として Target Objective ID を gObj-OB4-1
Activity2 の Primary Objective のマップ情報として Write Satisfied Status を true
Activity3 の Precondition Rule に If completed, then skip

Activity3 の Rollup Rules に Completed if all Satisfied
Activity4 の Primary Objective の ID を PRIMARYOBJ
Activity4 に Primary Objective のマップ情報として Target Objective ID を gObj-OB4-1
Activity5 の Primary Objective の ID を PRIMARYOBJ
Activity5 に Primary Objective のマップ情報として Target Objective ID を gObj-OB4-2
Activity6 の Primary Objective の ID を PRIMARYOBJ
Activity6 に Primary Objective の Objective Satisfied by Measure を true
Activity6 の Primary Objective の Minimum Normalized Measure を-0.75
Activity6 の Primary Objective のマップ情報として Target Objective ID を gObj-OB4-3

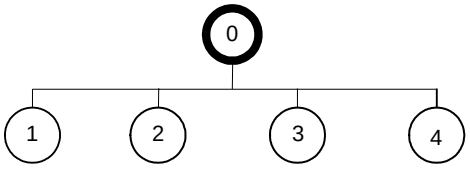
【動作】
Step2 で Activity1 の SCO から score.scaled に-0.25 をセットします。success_status はなにもセットされないため satisfied になります。Activity1 にマップされている共有学習目標 gObj-OB4-1 は Write Satisfied Status が true であることから satisfied sutetus は satisfied になります。gObj-OB4-3 は Write Normalized Measure が true であることから習得度は-0.25 となります。この時点では Activity3 の子 Activity のうち、gObj-OB4-1 のステータスから Activity4、gObj-OB4-3 の習得度から Activity6 のみが satisfied です。このため Activity3 の Rollup Rules は成り立たないため Activity3 は completed になっていません。

Step3 で Activity2 の success_status に passed をセットすることでマップされている共有学習目標の gObj-OB4-2 の satisfied sutetus は satisfied になります。gObj-OB4-2 のステータスから Activity5 が satisfied になり、これによって Activity3 の Rollup Rules が成り立って Activity3 が completed になります。そのため Continue リクエストに対して、Precondition Rule が成り立って Activity3 が Skip され Activity7 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: OB-5a

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Objective Satisfied by Measure == true Map Info: Target Objective ID == gObj-OB5a Write Satisfied Status == true Write Normalized Measure == true
2	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Objective Satisfied by Measure == true Minimum Normalized Measure == 0.4 Map Info: Target Objective ID == gObj-OB5a Sequencing Rules: Precondition Rule: If satisfied, then skip
3	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Minimum Normalized Measure == 0.1 Map Info: Target Objective ID == gObj-OB5a Sequencing Rules: Precondition Rule: If not satisfied, then skip
4	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery

2.	Set Activity 1's cmi.score.scaled to `0.85`; Set Activity 1's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 4 for delivery

【目的】
共有学習目標を使用した複数の Precondition Rule のテストを行います。

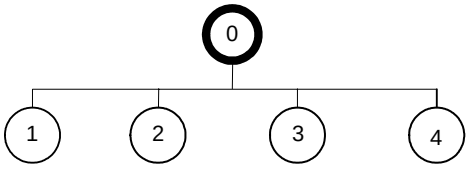
【構造】
共有学習目標として gObj-OB5a を使用
Activity1 の Primary Objective の ID を PRIMARYOBJ
Activity1 の Primary Objective の Objective Satisfied by Measure を true
Activity1 に Primary Objective のマップ情報として Target Objective ID を gObj-OB5a
Activity1 の Primary Objective のマップ情報として Write Satisfied Status を true
Activity1 の Primary Objective のマップ情報として Write Normalized Measure を true
Activity2 の Primary Objective の ID を PRIMARYOBJ
Activity2 の Primary Objective の Objective Satisfied by Measure を true
Activity2 の Primary Objective の Minimum Normalized Measure を 0.4
Activity2 に Primary Objective のマップ情報として Target Objective ID を gObj-OB5a
Activity2 の Precondition Rule に If satisfied, then skip
Activity3 の Primary Objective の ID を PRIMARYOBJ
Activity3 の Primary Objective の Minimum Normalized Measure を 0.1
Activity3 に Primary Objective のマップ情報として Target Objective ID を gObj-OB5a
Activity3 の Precondition Rule に If not satisfied, then skip

【動作】
Step2 で Activity1 の SCO から score.scaled が 0.85、success_status を `passed` とセットします。
この時点でマップされている共有学習目標の gObj-OB5a は Write Satisfied Status が true、Write Normalized Measure が true、Objective Satisfied by Measure で true、Minimum Normalized Measure が設定なしのためデフォルト値であることから、習得度は 0.85、satisfied sutetus は習得度との比較で failed となります。
これによって Activity2 の satisfied status の評価はマップされている共有学習目標 gObj-OB5a の習得度と比較し satisfid、Activity3 の satisfied status の評価は共有学習目標 gObj-OB5a のステータスから failed となり、Continue リクエストに対して Precondition Rule が成り立ち Skip されて Activity4 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: OB-5b

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-OB5b Write Satisfied Status == true Write Normalized Measure == true
2	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Objective Satisfied by Measure == true Minimum Normalized Measure == 0.75 Map Info: Target Objective ID == gObj-OB5b Sequencing Rules: Precondition Rule: If not objective status known , then skip
3	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Minimum Normalized Measure == 0.1 Map Info: Target Objective ID == gObj-OB5b Sequencing Rules: Precondition Rule: If satisfied, then skip
4	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Set Activity 1's cmi.success_status to `passed`;	Identify Activity 4 for delivery

	Process a *Continue* navigation request	

【目的】
共有学習目標を使用した複数の Precondition Rule のテストを行います。

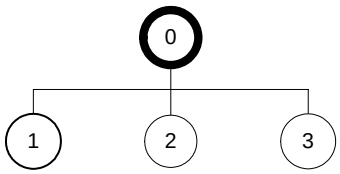
【構造】
共有学習目標として gObj-OB5b を使用
Activity1 の Primary Objective の ID を PRIMARYOBJ
Activity1 に Primary Objective のマップ情報として Target Objective ID を gObj-OB5b
Activity1 の Primary Objective のマップ情報として Write Satisfied Status を true
Activity1 の Primary Objective のマップ情報として Write Normalized Measure を true
Activity2 の Primary Objective の ID を PRIMARYOBJ
Activity2 の Primary Objective の Objective Satisfied by Measure を true
Activity2 の Primary Objective の Minimum Normalized Measure を 0.75
Activity2 に Primary Objective のマップ情報として Target Objective ID を gObj-OB5b
Activity2 の Precondition Rule に If not objective status known , then skip
Activity3 の Primary Objective の ID を PRIMARYOBJ
Activity3 の Primary Objective の Minimum Normalized Measure を 0.1
Activity3 に Primary Objective のマップ情報として Target Objective ID を gObj-OB5b
Activity3 の Precondition Rule に If satisfied, then skip

【動作】
Step2 で Activity1 の SCO から success_status を passed とセットします。この時点でマップされている共有学習目標の gObj-OB5b は Write Satisfied Status が true、Write Normalized Measure が true ですが score.scaled はセットされていないため習得度は unknown、satisfied sutetus は satisfied となります。
これによって Activity2 の習得度の評価はマップされている共有学習目標 gObj-OB5b から unknown、Activity3 の Satusfied status の評価は共有学習目標 gObj-OB5b のステータスから satisfied となり、Continue リクエストに対して Precondition Rule が成り立ち Skip されて Activity4 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: OB-5c

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false
1	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Objective Satisfied by Measure == true Minimum Normalized Measure == 0.0 Map Info: Target Objective ID == gObj-OB5c Write Satisfied Status == true Write Normalized Measure == true
2	Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-OB5c Sequencing Rules: Precondition Rule: If not objective status known , then skip
3	Default

Test Script:

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Set Activity 1's cmi.success_status to `passed`; Process a *Continue* navigation request	Identify Activity 3 for delivery

【目的】
共有学習目標を使用した Precondition Rule のテストを行います。

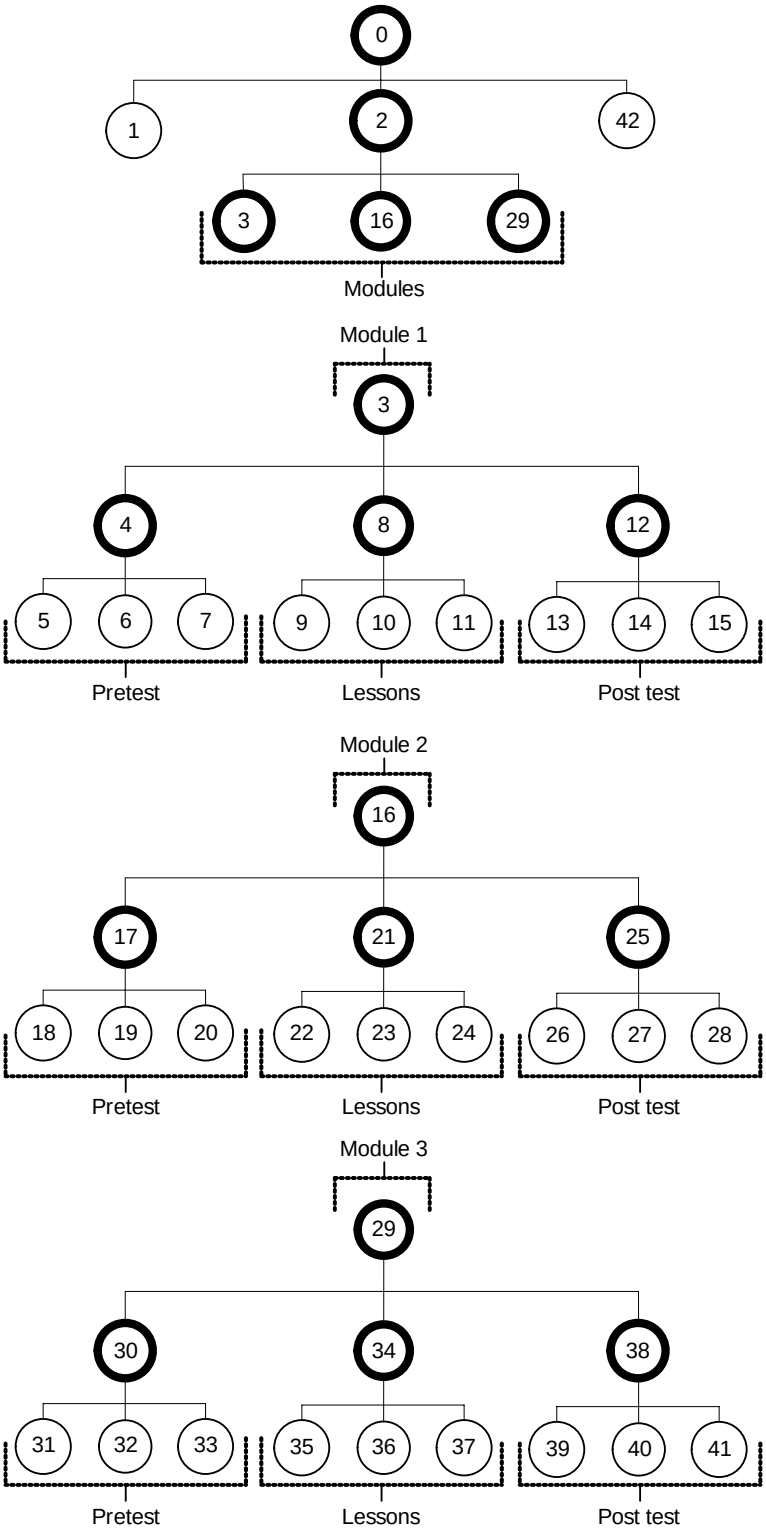
【構造】
共有学習目標として gObj-OB5c を使用
Activity1 の Primary Objective の ID を PRIMARYOBJ
Activity1 の Primary Objective の Objective Satisfied by Measure を true
Activity1 の Primary Objective の Minimum Normalized Measure を 0.0
Activity1 に Primary Objective のマップ情報として Target Objective ID を gObj-OB5c
Activity1 の Primary Objective のマップ情報として Write Satisfied Status を true
Activity1 の Primary Objective のマップ情報として Write Normalized Measure を true
Activity2 の Primary Objective の ID を PRIMARYOBJ
Activity2 に Primary Objective のマップ情報として Target Objective ID を gObj-OB5c
Activity2 の Precondition Rule に If not objective status known , then skip

【動作】
Step2 で Activity1 の SCO から success_status を passed とセットします。この時点でマップされている共有学習目標の gObj-OB5c は Write Satisfied Status が true、Write Normalized Measure が true、Minimum Normalized Measure が 0.0 ですが score.scaled はセットされていないため習得度は unknown、satisfied sutetus も unknown となります。
これによって Activity2 の習得度の評価はマップされている共有学習目標 gObj-OB5c から unknown となり、Continue リクエストに対して Precondition Rule が成り立ち Skip されて Activity3 が配信されます。

【補足】
Continue、Previous リクエストで前後に移動するために、中間ノードアクティビティのすべて Control Mode の Frow に True をセットしています。

Test Case: T-1

Activity Tree Structure:



Sequencing Information:

Activity	Sequencing Information
0	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If completed, then exit Rollup Rules: If any not satisfied, then not satisfied
1	Rollup Rules: Rollup Objective Satisfied == false
2	Control Model: Flow == true Choice == false Sequencing Rules: Exit Rule: If completed, then exit Post Condition Rule: If always, then continue
3	***Include Module Collection***
4	***Include Pretest Collection*** Objectives: Primary Objective: Objective ID == PRIMARYOBJ Objective Satisfied by Measure == true Minimum Normalized Measure == 0.6 Map Info: Target Objective ID == gObj-T1-1 Read Normalized Measure == false Write Satisfied Status == true Write Normalized Measure == true
8	***Include Lessons Collection*** Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-T1-1
12	***Include Posttest Collection*** Objectives: Primary Objective: Objective ID == PRIMARYOBJ Objective Satisfied by Measure == true Minimum Normalized Measure == 0.6 Map Info: Target Objective ID == gObj-T1-1 Write Satisfied Status == true Write Normalized Measure == true
16	***Include Module Collection***
17	***Include Pretest Collection*** Objectives: Primary Objective:

	Objective ID == PRIMARYOBJ Objective Satisfied by Measure == true Minimum Normalized Measure == 0.6 Map Info: Target Objective ID == gObj-T1-2 Read Normalized Measure == false Write Satisfied Status == true Write Normalized Measure == true
21	***Include Lessons Collection*** Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-T1-2
25	***Include Posttest Collection*** Objectives: Primary Objective: Objective ID == PRIMARYOBJ Objective Satisfied by Measure == true Minimum Normalized Measure == 0.6 Map Info: Target Objective ID == gObj-T1-2 Write Satisfied Status == true Write Normalized Measure == true
29	***Include Module Collection***
30	***Include Pretest Collection*** Objectives: Primary Objective: Objective ID == PRIMARYOBJ Objective Satisfied by Measure == true Minimum Normalized Measure == 0.6 Map Info: Target Objective ID == gObj-T1-3 Read Normalized Measure == false Write Satisfied Status == true Write Normalized Measure == true
34	***Include Lessons Collection*** Objectives: Primary Objective: Objective ID == PRIMARYOBJ Map Info: Target Objective ID == gObj-T1-3
38	***Include Posttest Collection*** Objectives: Primary Objective: Objective ID == PRIMARYOBJ Objective Satisfied by Measure == true Minimum Normalized Measure == 0.6 Map Info: Target Objective ID == gObj-T1-3 Write Satisfied Status == true Write Normalized Measure == true
42	Limit Conditions: Attempt limit == 1 Objectives: Primary Objective:

	Objective ID == PRIMARYOBJ Objective: Objective ID == obj1 Map Info: Target Objective ID == gObj-T1-1 Objective: Objective ID == obj2 Map Info: Target Objective ID == gObj-T1-2 Objective: Objective ID == obj3 Map Info: Target Objective ID == gObj-T1-3

Sequencing Collection	**Sequencing Information**
Module	Control Mode: Flow == true Choice == false Sequencing Rules: Exit Rule: If completed, then exit
Pretest	Control Mode: Flow == true Choice == false Forward Only == true Sequencing Rules: Precondition Rule: If satisfied, then skip Limit Conditions: Attempt limit == 1 Rollup Rules: Completed if all attempted Rollup Considerations: Required for Completion if attempted
Lessons	Control Mode: Flow == true Choice == false Sequencing Rules: Precondition Rule: If satisfied, then skip Rollup Rules: Rollup Objective Satisfied == false Rollup Progress Completion == false
Posttest	Control Mode: Flow == true Choice == false Forward Only == true Sequencing Rules: Precondition Rule: If satisfied, then skip Limit Conditions: Attempt limit == 1 Rollup Rules:

	Completed if all attempted Rollup Considerations: Required for Completion if not skipped

Test Script: T-1a

NOTE: This test script must be run independent from Test Script T-1b, so as not to have collisions of Global Shared Objective tracking information. Prior to running this Test Script, ensure that all global shared objective tracking information associated with Test Case T-1 (shared global objectives gObj-T1-1, gObj-T1-2, and gObj-T1-3) is set to the default (uninitialized) state.

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 5 for delivery
3.	Set Activity 5's cmi.score.scaled to `0.0`; Process a *Continue* navigation request	Identify Activity 6 for delivery
4.	Set Activity 6's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 7 for delivery
5.	Set Activity 7's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 18 for delivery
6.	Set Activity 18's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 19 for delivery
7.	Set Activity 19's cmi.score.scaled to `0.0`; Process a *Continue* navigation request	Identify Activity 20 for delivery
8.	Set Activity 20's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 31 for delivery
9.	Set Activity 31's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 32 for delivery
10.	Set Activity 32's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 33 for delivery
11.	Set Activity 33's cmi.score.scaled to `0.0`; Process a *Continue* navigation request	Identify Activity 42 for delivery

Test Script: T-1b

NOTE: This test script must be run independent from Test Script T-1a, so as not to have collisions of Global Shared Objective tracking information. Prior to running this Test Script, ensure that all global shared objective tracking information associated with Test Case T-1 is set to the default (uninitialized) state.

Step	Action	Expected Result
1.	Process a *Start* navigation request	Identify Activity 1 for delivery
2.	Process a *Continue* navigation request	Identify Activity 5 for delivery
3.	Set Activity 5's cmi.score.scaled to `0.0`; Process a *Continue* navigation request	Identify Activity 6 for delivery
4.	Set Activity 6's cmi.score.scaled to `0.0`; Process a *Continue* navigation request	Identify Activity 7 for delivery
5.	Set Activity 7's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 9 for delivery

6.	Process a *Continue* navigation request	Identify Activity 10 for delivery
7.	Process a *Continue* navigation request	Identify Activity 11 for delivery
8.	Process a *Continue* navigation request	Identify Activity 13 for delivery
9.	Set Activity 13's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 14 for delivery
10.	Set Activity 14's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 18 for delivery
11.	Set Activity 18's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 19 for delivery
12.	Set Activity 19's cmi.score.scaled to `0.0`; Process a *Continue* navigation request	Identify Activity 20 for delivery
13.	Set Activity 20's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 31 for delivery
14.	Set Activity 31's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 32 for delivery
15.	Set Activity 32's cmi.score.scaled to `1.0`; Process a *Continue* navigation request	Identify Activity 33 for delivery
16.	Set Activity 33's cmi.score.scaled to `0.0`; Process a *Continue* navigation request	Identify Activity 42 for delivery

【目的】
3 つのモジュールを含む大規模なアクティビティツリーで、共有学習目標を用いたシーケンシングが働くことをテストします。

【構造】
アクティビティツリーは、大きく Activity3, 16, 29 を親とする 3 つのモジュールから構成されています。
各モジュールは、それぞれ 3 つの Activity を含むプリテストクラスタ、レッスンクラスタ、ポストテストクラスタから構成されます。つまり、
Activity3 モジュール＝Activity4 プリテスト、Activity8 レッスン、Activity12 ポストテスト
Activity16 モジュール＝Activity17 プリテスト、Activity21 レッスン、Activity25 ポストテスト
Activity29 モジュール＝Activity30 プリテスト、Activity34 レッスン、Activity38 ポストテスト
です。
同一モジュール内のプリテスト、レッスン、ポストテストは、学習目標を共有しています。つまり、Activity3 モジュールでは gObj-T1-1、Activity16 モジュールでは gObj-T1-2、Activity29 モジュールでは gObj-T1-3 が共有されています。
プリテストでは Objective Satisfied by Measure を用いたロールアップが行われ、Minimum Normalized Measure 以上の習得度になると共有学習目標が satisfied になります。共有学習目標が satisfied になるとレッスン、ポストテストは Skip Precondition Rule でスキップされます。
プリテスト、ポストテストは Limit Condition により、一回しか試行できません。また、Forward Only が true に設定されており、元の問題に戻ることもできません。
プリテストは、Rollup Rules: Completed if all attempted と Rollup Considerations: Required for Completion if attempted が設定されていて、全部の子 Activity を試行すると Completed になり、また、試行した場合にはモジュールの Completion ロールアップの評価対象になります。
レッスンは、Rollup Objective Satisfied == false、Rollup Progress Completion == false のため、モジュールのロールアップの評価対象になりません。
ポストテストは Rollup Rules: Completed if all attempted、Rollup Considerations: Required for Completion if not skipped のため、全部の子 Activity を試行すると Completed になり、また、

Skip されなかった場合だけモジュールの Completion ロールアップの評価対象になります。
モジュールは、Exit Rule: If completed, then exit で、completed になると終了します。
Activity2 は Exit Rule: If completed, then exit、Post Condition Rule: If always, then continue ですので、completed になると終了し、終了すると必ず continue リクエストで前方に移動します。

【動作】
T-1a では、Step3, 4, 5 で Activity5, 6, 7 で、cmi.score.scaled が 0.0, 1.0, 1.0 に設定されて、Activity4 の学習目標習得度が 2/3 となって Minimum Normalized Measure 以上となるため共有学習目標 gObj-T1-1 が satisfied になります。
そのため、Step5 で、Activity8, 12 が Skip されて、Activity18 が配信されます。
Step5, 6, 7 および Step8, 9, 10 でも同様にプリテストが satisfied になり、レッスン、ポストテストが Skip されます。

T-1b では、Step3, 4, 5 で Activity5, 6, 7 で、cmi.score.scaled が 0.0, 0.0, 1.0 に設定されて、Activity4 の学習目標習得度が 1/3 となって Minimum Normalized Measure 以下となるため共有学習目標 gObj-T1-1 が not satisfied になります。
このため、Step5, 6, 7 でレッスンの Activity9, 10, 11 が配信され、Step8, 9 でポストテストの Activity13, 14 が配信されます。Activity13, 14 の cmi.score.scaled が 1.0, 1.0 となるため、Step10 で Activity3 の Exit が有効になり次のモジュールに入り、Activity18 が配信されます。
以降は T-1a と同じ動作になります。

【補足】