SCORM® 2004 3rd EDITION

Sharable Content Object Reference Model

Run-Time Environment

NOVEMBER 16, 2006
VERSION 1.0

Advanced Distributed Learning

このページは空白である.

Advanced Distributed Learning (ADL)

SCORM® 2004 3rd Edition

ランタイム環境 (RTE)
バージョン 1.0

ADLNet.gov から入手可能
(http://www.adlnet.gov/)

質問やコメントは ADL 問い合わせセンタ (ADLNet.gov) まで

このページは空白である.

日本語版訳者

eLC標準化推進委員会

伊藤義剛	株式会社BSNアイネット
大仲 輝	日立電子サービス(株)
熊澤 剛	(株)ヒューマンサイエンス
鈴木誠治	長岡技術科学大学
田中頼人	慶應義塾大学
樋田 稔	エスエイティーティー(株)
仲林 清	放送大学
中平勝子	長岡技術科学大学
本村孝則	アベイズム(株)
湯川浩一	(株)キバン

このページは空白である.

チーフ　テクニカル　アーキテクト
Philip Dodds

テクニカル　エディタ
Schawn E. Thropp

謝辞

ＡＤＬは，相互運用可能なイーラーニングの標準および仕様の作成に関する下記の組織およびメンバーの継続的，献身的な協力に感謝したい．

Alliance of Remote Instructional Authoring & Distribution Networks for Europe (ARIADNE) (http://www.ariadne-eu.org/)

Aviation Industry CBT Committee (AICC) (http://www.aicc.org/)

Institute of Electrical and Electronics Engineers (IEEE) Learning Technology Standards Committee (LTSC) (http://ltsc.ieee.org/)

IMS Global Learning Consortium, Inc. (http://www.imsglobal.org/)

ＡＤＬはＡＤＬコミュニティに関してもＳＣＯＲＭの進化への貢献に対して感謝したい．

Global Learning Consortium とそのメンバーや提供者はこの素材の利用によって実装者やサード・パーティに起こる直接または間接の被害に対していかなる義務も負わない.

著作権

配給

本書の配給を許可するには以下の条件を満たす必要がある:

1. 本書および本書に言及されたイメージおよび事例の利用は, 非営利目的で教育目的もしくは情報提供目的に限る.
2. 本書および本書に言及されたイメージおよび事例は, 修正されずそのままの形であること. これにはカバーページおよび著作権, 配給, 再生セクションも含まれる.

再生

本書を全てまたは部分的に再生するには以下の条件を満たす必要がある:

1. 再生は, 非営利目的で教育目的もしくは情報提供目的に限る.
2. 情報元として以下を適切に引用する:

   情報元:Advanced Distributed Learning (ADL), Sharable Content Object Reference Model (SCORM®) 2004 3rd Edition Run-Time Environment Version 1.0, 2006.

著作権, 配給および再生に関するこれ以上の情報および質問には, 下記までお問い合わせ下さい:

ADL Co-Laboratory Hub
1901 North Beauregard Street, Suite 600
Alexandria, Virginia 22311
USA
703-575-2000

このページは空白である．

目次

セクション 1
SCORM ランタイム環境　（RTE）

このページは空白である．

1.1. SCORM ランタイム環境（RTE）ブック概説

Sharable Content Object Reference Model(SCORM)は，しばしば本棚の一組の本に例えられる．ランタイム環境 (RTE) ブックは，その一組の本の中の一冊である．(図 1.1.a：SCORM ブックシェルフの 1 部としてのランタイム環境ブック参照) ．他のブックおよびブック間の関係についてのより詳細な情報は，SCORM2004 概要に記述されている．SCORM RTE ブックは，ランタイム環境管理における LMS(Learning Management System)の要件を記述している (すなわち，コンテンツ起動プロセス，コンテンツと LMS 間の標準化された通信，そして学習者のコンテンツとの経験に関する情報の伝達に利用される標準データモデル要素通信) ．RTE ブックは，SCO(Sharable Content Objects)の要件，および，SCO による共通 API(Application Programming Interface)と SCORM ランタイム環境データモデルの利用も扱う．

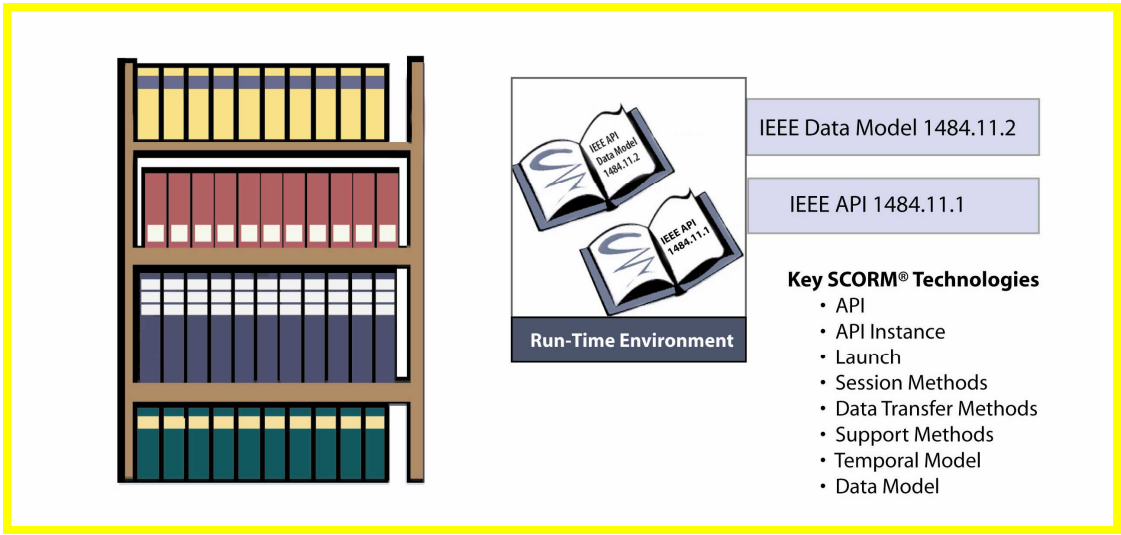

図 1.1.a：SCORM ブックシェルフの 1 部としての SCORM ランタイム環境ブック

1.1.1. SCORM ランタイム環境ブックで取り上げられる内容は？

SCORM RTE ブックでは，いくつかの主要コンセプトが導入されている．このブックは，ランタイム時にコンテンツオブジェクト (SCO もしくはアセット) をシーケンシングすること，および，SCO にナビゲーション要求を提示させることに関する LMS の基本的な役割を取り上げている．さらに，学習者にナビゲーションコントロールを提供するためにガイダンスが提供される．議論される一般的な課題には以下が含まれる：

- ランタイム環境管理: コンテンツオブジェクト– SCO およびアセット－の起動．SCO との通信管理，ランタイム環境データモデル管理通信
- API (Application Programming Interface)： LMS API 要件，SCORM 通信要件，通信エラー条件
- ランタイム環境データモデル： データモデル管理および動作要件，データタイプ要件

1.1.2. SCORM RTM ブックの利用

本書は，自社製品を SCORM 対応にしようとする LMS，SCO および編集ツールベンダ，そして，コンテンツと LMS 間の通信関係を理解したい SCORM コンテンツ開発者などに役立つであろう．

本書の「セクション 1 The SCORM Run-Time Environment (RTE)」および「セクション 2 Managing the Run-Time Environment (RTE)」は，一般的な RTE 関連の概念を取り上げる．これらのセクションは，SCORM RTE の概略を知りたい人および技術的な深堀りをしたくない人に薦められる．他には，RTE の最新情報を知りたい人などに役立つと思われる．例えば，「セクション 2.1 ランタイム環境 (RTE) 管理」は，新たなシーケンシングおよびナビゲーションブックがどのように SCORM RTE に影響を与えるかについて述べている．

「セクション 3 Application Programming Interface (API)」は，RTE に関する技術的な詳細を提供する最初のセクションである．このセクションは，コンテンツ開発者が利用可能なすべての SCORM API 関数およびエラーメッセージを説明する．また，サンプルコードおよび API 利用要件およびガイドラインも提供する．

「セクション 4 SCORM ランタイム環境データモデル」は，すべての SCORM データモデル要素の詳細を取り扱う．これには，ある要素に関連した LMS および SCO の特定の動作要件のリストを含む．

1.1.3. 他の SCORM ブックとの関係

それぞれの SCORM ブックは単体でも活用できるように意図されている一方で，内容が互いに重複する部分がある．例えば，本書は，主にコンテンツおよび LMS 間の通信に焦点を当てるが，その通信を行う SCO(Sharable Content Objects)に頻繁に言及する．SCO は CAM ブックでも詳細に説明される．

同様に，SN ブックは，シーケンシングおよびナビゲーションプロセスの詳細を取り上げ，これには，LMS が，ナビゲーション要求および関連するアクティビティをどのように評価するかの詳細を含むが，本書では，コンテンツ配信を取り上げているため，LMS が配信するコンテンツをどのように決定するかについては，簡単に触れるに留めている．

1.1.3.1 SCORM コンテンツアグリゲーションモデル(CAM) ブック

SCORM CAM ブック [18] は，コンテンツアグリゲーションを構築するための役割と要件(コンテンツの収集，ラベリングおよびパッケージングプロセス)を説明する．そこでは，メタデータ，コンテンツパッケージ，および ADL シーケンシング＆ナビゲーションのコンテンツパッケージ拡張に関する情報が含まれる．SCORM CAM ブックと SCORM SN ブックは相互に関連している部分がいくつかある．

メタデータは，「データに関するデータ」である．メタデータは SCORM コンテンツモデルの異なる構成要素（コンテンツアグリゲーション，コンテンツ構成，アクティビティ，SCO およびアセット）を記述する．メタデータはラベリングの一種で，これらの構成要素の検索，発見を容易にする．現時点で，メタデータと SCORM ランタイム環境データモデルの間に定義された関係はない．これらの理由により，メタデータについて，SCORM RTE ブックでは詳細に議論されていない．この関係は，SCORM の進化によって変わる可能性がある．

コンテンツパッケージは，一般的な意味で，マニフェストに記述されたコンテンツ構成で，コンテンツオブジェクトを集約する．SCORM コンテンツパッケージは，SCORM コース，レッスン，モジュール，もしくは，単にリポジトリに保存された関連するコンテンツオブジェクトの集合を指すことがある．マニフェスト，すなわちすべての SCORM コンテンツパッケージの本質的な部分は，imsmanifest.xml と呼ばれる Extensible Markup Language(XML)ベースのファイルを含む．このファイルは，多くの点で，荷札に似ていて，パッケージの内容を表し，オプションでコンテンツ構成の説明を含む．

SCORM コンテンツパッケージは，LMS が，どのようにコンテンツパッケージやその内容を処理するかを表す追加情報を含むことがある．この情報のいくつかは SCORM RTE モデルで利用されている．

- コンテンツオブジェクト起動ロケーションおよび起動パラメータは，SCORM コンテンツパッケージにおける要素として記述されている．これらの要素は，コンテンツオブジェクトの起動や配信に必要不可欠である．SCORM ランタイム環境ブックは，これらの要素およびコンテンツオブジェクトの起動へのこれらの要素の影響を詳述する．
- SCORM コンテンツパッケージのいくつかの要素は，コンテンツオブジェクトのランタイムデータモデルの初期化および管理に影響がある．SCORM RTE ブックは，これらのデータモデル要素および要求される LMS 動作について詳述する．
- SCORM コンテンツパッケージの他の要素は，コンテンツオブジェクトランタイムデータモデルの特定の要素の初期値を記述する．SCORM RTE ブックでは，これらのデータモデル要素および初期化動作を詳述する．
- SCORM コンテンツパッケージが，コンテンツ構成の記述を含むとき，パッケージのコンテンツオブジェクトのシーケンシングに関する意図されたアプローチを定義するのに,シーケンシング情報が追加されることがある．SCORM コンテンツパッケージは，ある特定の UI ナビゲーションコントロールが,どのように提示されるか，実行可能にされるかもしくは隠されるかに関してのガイダンスを LMS に提供するためのユーザインターフェース (UI) 要素を含むことがある．コンテンツオブジェクトが，このブックで定義されたように起動されると，これらの要素は，シーケンシング情報と連動して (SCORM シーケンシングおよびナビゲーションブック参照)，(表示の際に) 正しい UI ナビゲーションコントロール(例えば“continue” もしくは“previous”ユーザーインターフェースコントロール)を提示するために利用される事がある．

SCORM コンテンツパッケージで上記のすべての要素がどのように指定されるか，より深く理解したい場合，SCORM CAM ブックを参照されたい．

1.1.3.2 SCORM シーケンシングおよびナビゲーション(SN) ブック

SCORM シーケンシングおよびナビゲーション (SN：Sequencing and Navigation) ブックは，IMS シンプルシーケンシング (SS：Simple Sequencing) 仕様 [17]に基づく．IMS SS 仕様は，どんな LMS でも一貫性のある方法で分割された学習アクティビティをシーケンスするという作成された学習経験の意図された動作を表す方法を定義する．

SCORM SN モデルは，どのように IMS SS が SCORM 環境で適用され，拡張されるかを定義する．SCORM 対応 LMS がランタイム時にシーケンシング情報を処理するために，実装しなければならない必要な動作と機能性を定義する．より具体的には，起動されたコンテンツオブジェクトおよび作成されたシーケンシング戦略と学習者との相互作用の結果に基き，アクティビティツリーにおける学習アクティビティの分岐とフローを記述する．アクティビティツリーは，LMS が各学習者に対して管理する学習アクティビティの概念的構造である．

SCORM SN ブックは，学習者やシステム主導によるナビゲーションイベントがどのように発生し，処理されうるのか，そして，結果として配信される学習アクティビティがどのように決められるかを説明している．配信されることが決められた各学習アクティビティは，関連したコンテンツオブジェクトを持っている．SCORM RTE モデルは，配信されるコンテンツオブジェクトがどのように起動されるかを記述している．ある学習者とコンテンツ構成にとって，起動したコンテンツオブジェクトのシーケンスは，学習活動 (コンテンツオブジェクトとの学習者の相互作用) を提供する．SCORM RTE モデルは，LMS が学習活動の結果をどのように管理するか，および，その学習活動がアクティビティツリーにどのように影響するかを記述する．

1.2. ランタイム環境概要

本書は SCORM RTE モデルを定義している. そこでは, コンテンツオブジェクトの起動, LMS と SCO 間の通信の確立, および, LMS と SCO の間で通信される追跡情報の管理に関する要件が詳述されている. SCORM のコンテクストでは, コンテンツオブジェクトは以下の 2 つのうちどちらかである:

- SCO(Sharable Content Objects):ランタイム時に通信する
- アセット:ランタイム時に通信しない

SCORM RTE ブックは, 共通のコンテンツオブジェクト起動メカニズム, コンテンツオブジェクトと LMS 間の共通の通信メカニズム, 学習者のコンテンツオブジェクトとの学習経験を追跡する共通のデータモデルを記述する. これらの観点は, いくつかの ADL 高度要件を満たす環境を形成する. 例えば, 標準化された通信メカニズムを通して通信するコンテンツオブジェクトは, ある LMS から他の LMS へ通信試行の修正を行わずに移行することが可能である. これは, 学習オブジェクトの移動性, 耐用性を増し, それにより開発・実装・保守コストを低減する.

SCORM RTE は, ある特定のコンテンツオブジェクトが起動されるように指定された時点を取り上げるモデルを定義する. 起動されるコンテンツオブジェクトの実際の特定方法については, 本ブックの対象範囲外であり, SCORM SN ブック[11]で展開されている.

本ブックは, ランタイム環境の管理だけを取り扱い, 以下が含まれる:

- コンテンツオブジェクトの学習者の Web ブラウザへの配信 (起動),
- 必要であれば, どのようにコンテンツオブジェクトが LMS と通信するか, および
- コンテンツオブジェクトに対して, どの情報が追跡されるかおよびどのように LMS がその情報を管理するか

以下のセクションは, SCORM RTE ブックと他の SCORM ブックの関係を説明する. さらに, 本ブックを理解するのに, SCORM 全体の専門家にならなければいけないということがないように, 良く使われる用語を紹介する. しかし, これは SCORM およびそのコンセプトを学び, 適用するには, 効果的な方法ではない. SCORM の各ブックを読み, すべての SCORM のコンセプトの目的, 詳細, 関係および優位性を良く理解することが強く推奨される.

SCORM は複数の LMS にわたって再利用可能および相互運用可能となるコンテンツオブジェクトを実現するために開発された. これを可能にするには, コンテンツオブジェクトを起動, 管理する共通の手段, コンテンツオブジェクトが LMS と通信するための共通の機構, および通信の基礎をなすあらかじめ定義された言語・語彙などが必要である. 図 1.2a に示された通り, RTE の三つの側面は, 起動, API, およびデータモデルである.

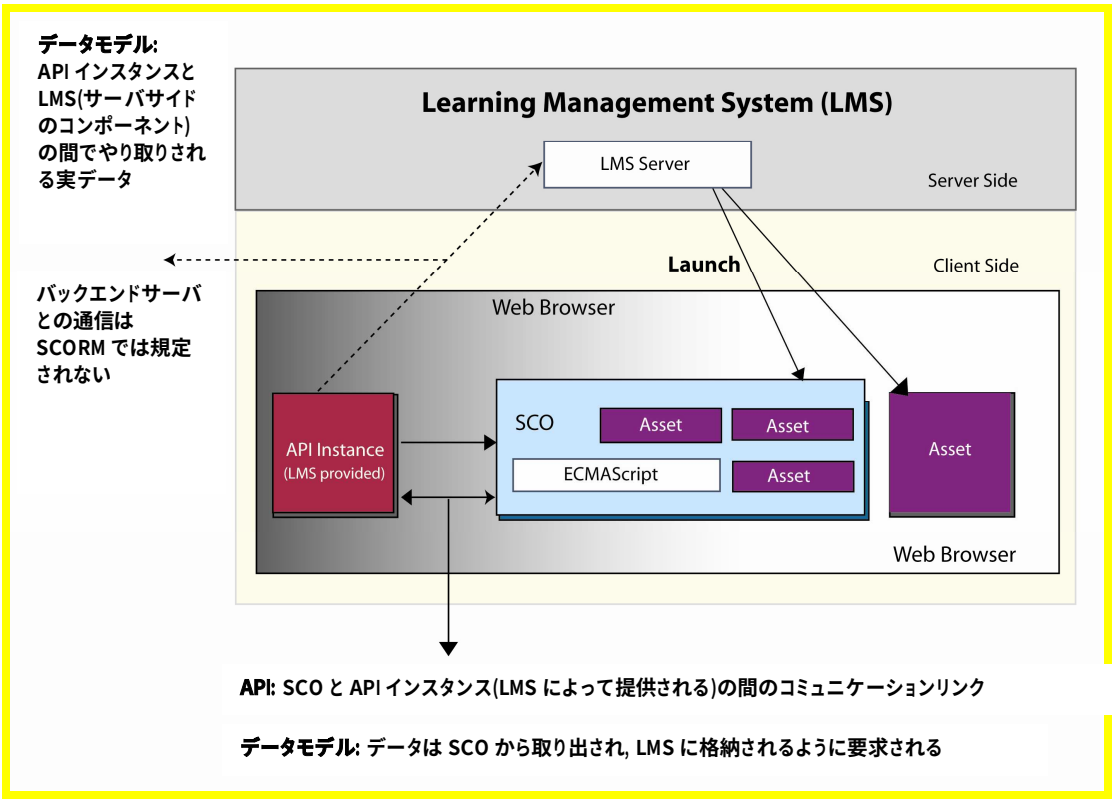

図 1.2a: SCORM の概念的ランタイム環境

起動プロセスは, LMS が Web ベースのコンテンツオブジェクトを開始するための共通の手段を定義する. 「コンテンツオブジェクト」という用語は, 学習者に対して起動されるあらゆるコンテンツを表すための総称的な用語として用いられる. SCORM では, 2 種類のコンテンツオブジェクト, SCO とアセット, がある. 起動プロセスは, 起動されたコンテンツオブジェクトと LMS の間の通信を確立する手順と役割を定義する. 通信メカニズムは共通 API により標準化される.

API は, コンテンツオブジェクトと LMS 間の概念的な通信状態(初期化された, 終了された, および/もしくは, エラー条件など)を LMS に知らせるための通信メカニズムであり, また, LMS および SCO の間でデータ(例えば, 得点, 時間制限)の取り出しと格納に利用される.

データモデルは, SCO の完了状態, もしくは, クイズやテストなどの評価からの得点といった, SCO を追跡するための情報を規定するのに利用されるデータモデル要素の標準セットである. 最も単純な形では, データモデルは, LMS および SCO 両者が「知っている」データモデル要素を定義する. LMS は, 複数の学習者セッションをまたがって SCO のデータモデル要素の状態を保持しなければならない. SCO は, 複数のシステムにまたがって確実に再利用されるために, これらのあらかじめ定義されたデータモデル要素だけを利用しなければならない.

このページは空白である．

セクション２
ランタイム環境（RTE）の管理

このページは空白である．

2.1. ランタイム環境 (RTE)管理

学習者がコンテンツオブジェクトとやり取りしているとき（学習体験），LMS は，学習者のパフォーマンスおよびナビゲーション要求を評価する（SCORM SN ブック参照）．LMS が，学習者へ配信するアクティビティを特定する時，アクティビティはそれに関連するコンテンツオブジェクトを持つ．LMS は，コンテンツオブジェクトを起動し，学習者に提示する．図 2.1a は，コンテンツ構造（manifest の organization セクション）が，アクティビティツリーで，どのように解釈されるかを示している．ツリー表示はマニフェストで見られるコンテンツ構造を異なった形で提示しているだけである（SCORM CAM 参照）．

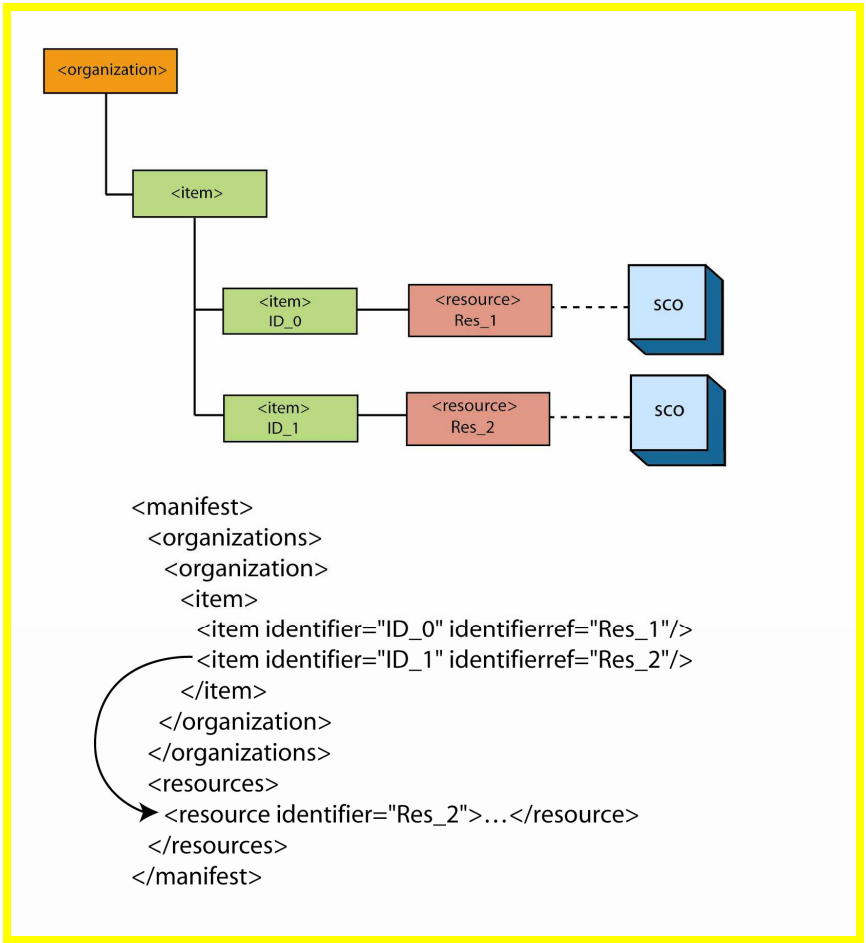

図 2.1a: コンテンツオブジェクトの起動

共通起動モデルは，学習経験のコンテクスト内で SCO およびアセットの形で Web ベースのコンテンツオブジェクトの配信を行う．この起動モデルは，LMS を通したコンテンツオブジェクト起動動作を，基になる LMS 実装を特定することなく一貫性のあるものにする．このコンテクストで，用語「LMS」はコンテンツオブジェクトの起動を提供するあらゆるシステムに適用される．

2.1.1. ランタイム環境時間モデル

関連するコンテンツオブジェクト (SCO ないしアセット) を有するアクティビティが配信されるように特定され，コンテンツオブジェクトが，学習者のブラウザ環境で起動されると，学習者は，コンテンツオブジェクトに取り組んだことになる．学習経験中，学習者の追跡を可能とするために，いくつかのカギとなる観点が定義される必要がある．以下の用語が IEEE 1484.11.1 Draft Standard for Learning Technology – Data Model for Content Object Communication [1]によって定義されおり，このドキュメントを通して参照される．

学習者試行 (Learner Attempt) – コンテンツオブジェクトを用いた学習アクティビティの要件を達成するための学習者の追跡される試行．試行は，一つ以上の学習者セッションに渡ることがあり，これらのセッションの間に，中断されることもある[1]．

学習者セッション (Learner Session) – 学習者がコンテンツオブジェクトにアクセスしている中断されない期間 [1]

通信セッション (Communication Session) – コンテンツオブジェクトと API (Application Programming Interface) 間の活性化された接続 [1]

ログインセッション (Login Session) – 学習者がシステムでセッションを開始(logged on)してから終了(logged out)するまでの間の時間帯

これらの 4 つの用語は，SCO のランタイム環境を管理するときに意味がある．アセットに対しては，RTE は，独立した学習者試行および学習者セッションからのみ成り立つ．各アセットの起動に対して，一学習者試行とこれに対応する学習者セッションだけである．一旦配信するアクティビティが特定されると，学習者試行が始まる (SCORM SN ブック参照) ．この試行中，学習者はコンテンツオブジェクト (SCO もしくはアセットどちらか) に取り組む．一旦学習者が取り組むと (コンテンツオブジェクトが，学習者の Web ブラウザで起動されると) ，学習セッションが始まる．起動されたコンテンツオブジェクトが，SCO であれば，SCO が LMS との通信を初期化すると，通信セッションが始まる．通信セッションは，SCO が LMS との通信を終了すると，終了する．学習セッションは，SCO を中断状態にしても終了することができるし，SCO を正常の状態で抜けても終了することができる．SCO にとって，学習者アテンプトは，学習セッションが正常な状態で終わったときに，終了する．アセットにとって，学習者アテンプトは，一旦アセットが学習者から離されたら，終了する．図 2.1.1a は，これらの 4 つの用語とこれらの用語の特定の SCO に対する関係を示す．

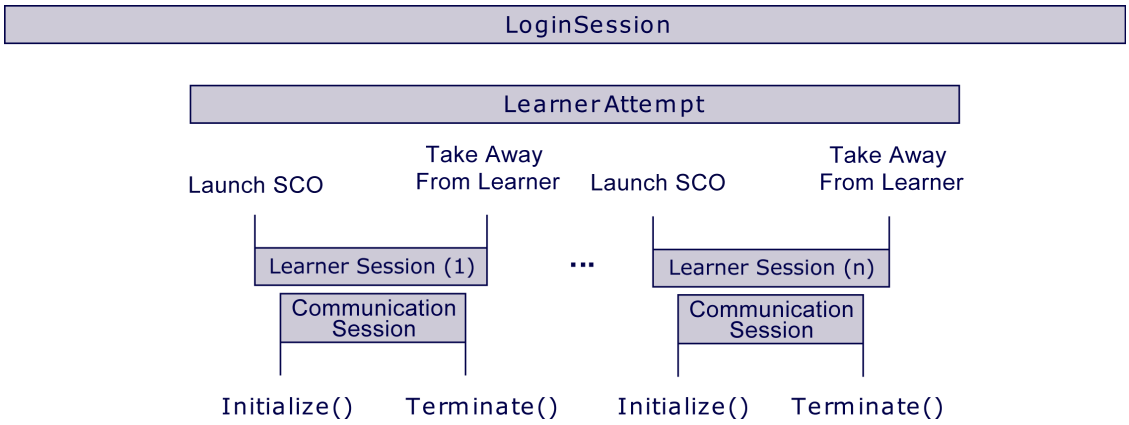

図 2.1.1a: 特定の SCO に対する時間モデル関係

2.1.1.1 学習者試行と学習者セッションの管理

学習者試行は，SCO が LMS と通信するランタイムデータモデル要素の管理方法を規定する重要な LMS 要件と関連している．SCO に対して新たな学習者試行が始まる際に，LMS は，SCO がアクセスし

利用するランタイムデータの新たなセットを作成し初期化しなくてはならない (セクション 4: *SCORM ランタイム環境データモデル*参照) . SCORM は, 新たなランタイムデータのセットが, いつ作成されるか定義しないが, 全てのデータモデルアクセスが, 新たなランタイムデータに対して行われているように見えなければならない.

LMS が前回の試行のデータをどう扱うかについては, SCORM の対象範囲外である. LMS は, このデータを, 履歴もしくは報告, 監査, 分析などの目的で格納することを選択できる. LMS は, 前回の試行中に集めたランタイムデータを, 削除することもできる. LMS に対する唯一の要求は, 学習者が試行を中断した際, 学習試行に対するランタイムデータを保持することだけである. SCORM は, 前回の学習者試行データの保持に関する LMS 要件を定義しない. しかし, 学習者セッションが中断され, 学習者試行が未完了になったら, SCO に対する次のセッションが開始するとき, 中断前に設定されたランタイムデータが利用可能であることを LMS は保証しなくてはならない. SCO と関連する学習アクティビティが次に配信対象として特定されると, 前回の学習者試行が再開され, 前回の学習者試行からのランタイムデータが, 新たな学習セッションに提供される.

以下の図(2.1.1.1a, 2.1.1.1b and 2.1.1.1c)は, 学習者試行と学習者セッション間の様々な関係を示す. 図 2.1.1.1a は, 一学習者セッションで単一の学習者試行が達成された場合を示す.

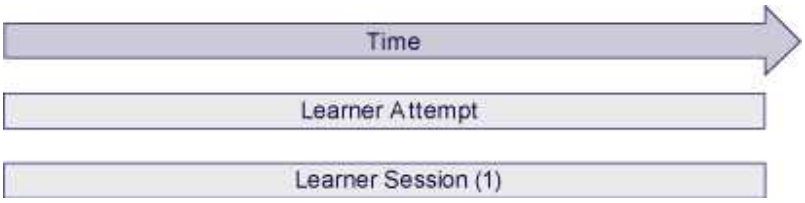

図 2.1.1.1a: 単一学習者アテンプトと 1 学習セッション

図 2.1.1.1b は, 単一の学習者試行が, いくつかの学習者セッションに渡る場合を示す. これらのセッションは中断され, 学習者セッションが正常な状態で終わるまで, 学習者試行は再開される.

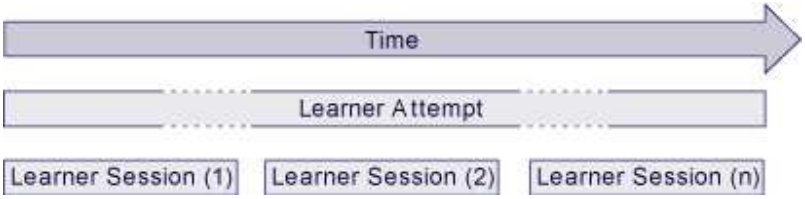

図 2.1.1.1b: いくつかの学習者セッションに渡る 学習者アテンプト

図 2.1.1.1c は, 連続的な複数の学習者試行を示す. 各学習者試行内で, 学習者セッションはいくつでもおこることが可能である.

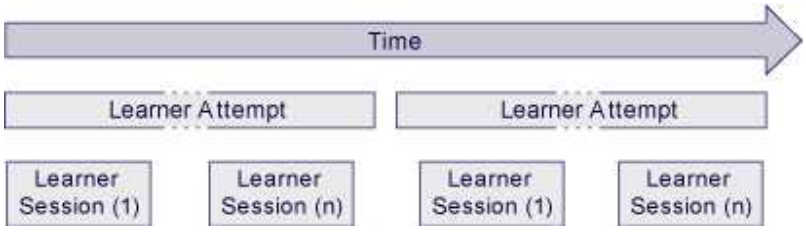

図 2.1.1.1c: 継続的な学習者アテンプト, 各々はいくつかの学習者セッションに跨る

2.1.1.2 学習者試行およびアクティビティをまたがるランタイムデータの保持

いくつかのケースでは，学習アクティビティに関連付けられた SCO への全ての学習者試行にわたって，学習アクティビティが，一組だけのランタイムデータだけを持つ必要がある．この要件は，SCO リソースが，試行と試行の間状態（ランタイムデータ）を保持する必要があると宣言することで明示される（この能力の定義の詳細は，SCORM CAM ブック参照）．ランタイムデータが全ての学習アテンプトの間で保持されると，学習アクティビティが定義した場合，その学習アクティビティに関連付けられた SCO への最初の学習者セッションが始まるときだけ，LMS はランタイムデータを作成して初期化を行うべきである．状態の保持はアセットに関連付けられたアクティビティに影響しない．

いくつかのケースでは，二つ以上の学習アクティビティが，同一の SCO リソースを参照することがある．その SCO リソースが，学習者アテンプト間で，その状態を保持しなければならないと定義すれば，その状態は，その SCO リソースを参照する学習アクティビティ間でも保持される．図 2.1.1.2a では，二つのアクティビティ(アイテム A12 およびアイテム A51 であらわされている)が，同一の SCO リソース(R2)を参照している．SCO リソースは，保持状態が True と定義されているので，LMS は，SCO への学習者試行の間のランタイムデータを保持しなければならない（2 つのアクティビティのどちらかで）．アクティビティ A12 の学習者試行間に，ランタイムデータが，SCO によって設定されたら，この同一のデータは，その後のアクティビティ A51 への学習者試行によってアクセスされる（すなわち，学習者試行とアクティビティをまたがって状態を保持する）．

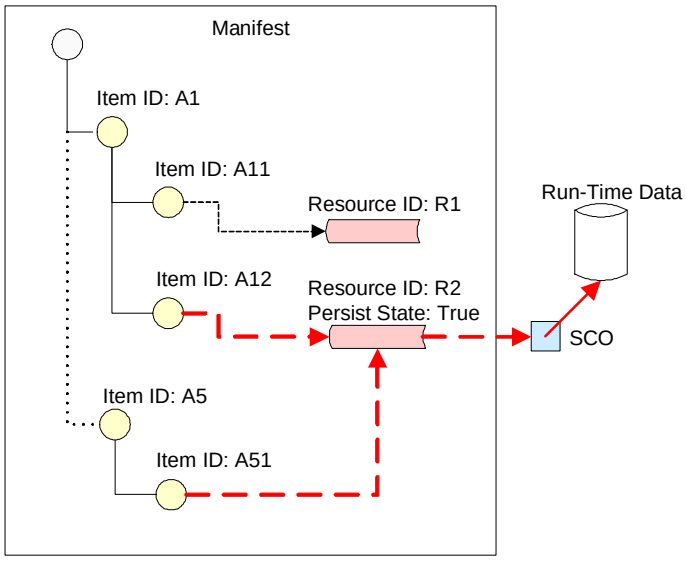

図 2.1.1.2a: 学習者アテンプトとアクティビティを跨る状態の保持

図 2.1.1.2b では，二つのアクティビティ（アイテム A12 およびアイテム A51）が，同一の SCO リソース(R2)を参照している．しかし，SCO リソースは，保持状態を False と定義された（何も定義されていないならば，デフォルト値）ので，LMS は，各アクティビティの間，SCO への学習者試行に対して，新規のランタイムデータを作成しなければならない．例えば，アクティビティ A12 への学習者試行の間，ランタイムデータが，SCO によって設定されたら，このデータは，学習者試行およびアクティビティに跨って保持されない．これは，アクティビティ A51 への試行中の SCO への学習者試行の間，A12 への試行中に設定されたデータは，利用できないということを意味する．

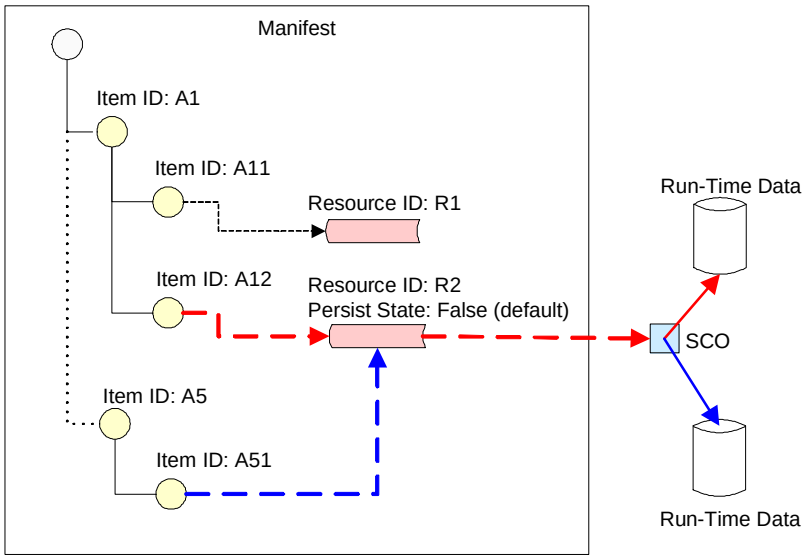

図 2.1.1.2b: 学習者アテンプトとアクティビティを跨いだ状態保持ができない

2.1.2. コンテンツオブジェクトの起動

コンテンツアグリゲーションモデル[19]で定義されたように，SCORM コンテンツモデルは 3 つの部品から成り立つ：

- アセット
- SCO
- コンテンツオーガニゼーション

SCORM CAM ブックは，起動可能なコンテンツオブジェクトの特徴を記述している．SCO およびアセットは起動可能な定義されたコンテンツモデル部品である．コンテンツオブジェクトのタイプによって，異なった起動要件が，存在する．起動プロセスは，LMS がコンテンツオブジェクトを学習者の Web ブラウザに起動する共通の方法を定義する．起動されたコンテンツオブジェクトと LMS 間の通信を確立する手順と役割は，起動されたコンテンツオブジェクトのタイプによって異なる．

定義された動作およびアクティビティに適用されたシーケンシング情報の評価に基づいて，学習アクティビティ間のシーケンシングを管理するのは，LMS の役割である（SCORM SN ブック参照）．学習経験を構成する学習アクティビティを通じた進捗は，定義されたシーケンシング情報および学習者とコンテンツオブジェクトの相互作用に応じて，シーケンシャルにもノンシーケンシャルにも，ユーザー主導型にもユーザー適用型にもなる．

ナビゲーションイベントに基づき，どの学習アクティビティを配信するか決定するのは LMS（ないしは，ここではシーケンシング要素／サービス）の役割である．LMS は，コンテンツ構造で定義された順序で，次の学習アクティビティを特定することもあれば，ユーザー選択の学習アクティビティを特定することもあり，また，以前経験したコンテンツオブジェクトの学習者パフォーマンスに基づいて，配信するアクティビティを適応的に決定することもある．配信へ識別された学習アクティビティは，常に関連するコンテンツオブジェクトを持つ．特定された学習アクティビティに関連付けられたコンテンツオブジェクトを起動するのは，LMS（ないしは，ここでは起動要素／サービス）の役割である．起動するコンテンツオブジェクトを決定すると，LMS はコンテンツパッケージに定義された（コード例 2－1 参照）コンテンツオブジェクトの起動ロケーションによって定義された URL(Universal Resource Locator)を用いて，現在表示されたコンテンツオブ

ジェクトを起動ロケーションで指定されたコンテンツオブジェクトに切り替える，つまりナビゲートするのに利用する．

```
<manifest>
   <organizations>
      <organization>
         <item>
            <item identifierref="RES_1">…</item>
            <item> … </item>
            <item> … </item>
         </item>
      </organization>
   </organizations>
   <resources>
      <resource identifier="RES_1"
                type="webcontent"
                adlcp:scormType="sco"
                href="Lesson1/Module1/sco1.htm"> … </resource>
   </resources>
</manifest>
```

コード例 2-1: 起動に利用される Href

適切な絶対 URL(fully qualified URL)を決定するのは LMS の役割である．SCORM CAM は，IMS コンテンツパッケージング仕様[18]を利用して，どのように絶対 URL を作成するかについての要件を定義する．URL は（マニフェストに存在すれば）以下の要素に基づいて作成される：

- xml:base declarations
- 起動パラメータ (URL のクエリ要素[6])
- Href 宣言

コンテンツオブジェクトに対する絶対 URL の構築に含まれるプロセスの詳細情報」については，SCORM CAM を参照されたい．

LMS は，起動をどのようにでも実装することができ，もしくは，必要に応じてクライアントまたは LMS のサーバ部分に実際の起動の役割を託すこともできる．実際の起動は，HTTP（Hypertext Transfer Protocol）を利用して行われなければならない．最終的には，コンテンツパッケージの起動ロケーションによって指定されたコンテンツオブジェクトは，起動されクライアントの Web ブラウザに配信される．

2.1.2.1 アセット

アセットを表すコンテンツオブジェクトに対して，SCORM 起動モデルは，LMS が HTTP プロトコルを利用してアセットを起動することしか要求しない．アセットは，API もしくはデータモデルを用いて LMS と通信しない．

2.1.2.2 Sharable Content Object (SCO)

SCO を表すコンテンツオブジェクトに対して，SCORM 起動モデルは，LMS が一度に（学習者毎に）一つの SCO を起動，追跡することを要求する．言い換えると，LMS は一度に（学習者毎に）一つの SCO を起動，追跡する．

起動された SCO は，それ自身が起動，追跡する従属 SCO に対して API インスタンスを実装できる．LMS は，これらの従属 SCO を知る責任はない．この場合，LMS によって起動された SCO は，自身が終

了するとき，全てのクリーンアップ（従属 SCO のために開かれた全ての画面を閉じること）に責任があることになる．LMS との全ての通信は，LMS に起動された SCO と行わなければならない．図 2.1.1.2a は，上記のシナリオを示す．SCO 1 は，LMS によって起動され，LMS は SCO 1 との通信セッションをどのように行うかわかっている．SCO 1 は SCO 1.1 を起動する何らかのメカニズムを持っている．SCO 1 は，API インスタンスの実装を有している．SCO 1.1 と SCO 1 間の全ての通信は，お互いの間で起こる．SCO 1 は，LMS が提供する API インスタンスへの参照を利用して，LMS に他のどんなデータを通信するかしないか決めることができる．SCO 1.1 が，LMS が提供する API インスタンスを通して直接 LMS に通信することは許されていない．

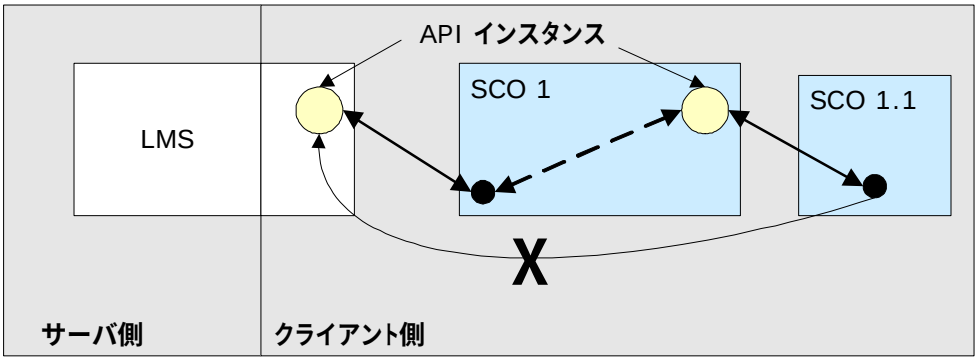

図 2.1.2.2a: SCO および従属 SCO

LMS は，API インスタンスを Document Object Model (DOM)オブジェクト[8]として提供する独立したブラウザ画面（ポップアップ画面）ないし LMS 画面の子フレームの中に SCO を起動しなければならない．API インスタンスは LMS が提供しなくてはならない．

API インスタンスが見つかるまで，親および/もしくは opener 画面階層を再帰的に検索するのは，SCO の役割である．一旦 API インスタンスが見つかったら，SCO は LMS との通信を開始できる．「セクション 3.2 SCO の責任」では，SCO の要件を定義する．SCO は API インスタンス機能（「セクション 3.1 Application Programming Interface」参照）で定義された全ての要件を遵守する責任がある．

2.1.3. コンテンツオブジェクトの終了

学習者セッションが終了する際に，現在学習者が経験しているコンテンツオブジェクトは取り除かれ，配信に特定された次のコンテンツオブジェクトと置き換えられる（セクション 2.1.2：コンテンツオブジェクトの起動参照）．一般的にコンテンツオブジェクトは，学習者もしくはシステム主導のナビゲーションイベントによって取り除かれる（SCORM SN ブック参照）．現在のコンテンツオブジェクトが取り除かれたあと，LMS は，正しいシーケンシング評価をするために，学習者のコンテンツオブジェクトとのやり取りに関する最も正確な情報を必要とする．取り除かれたコンテンツオブジェクトがアセットなら，LMS は，学習者の相互作用に関する仮定を行う．もし取り除かれたコンテンツオブジェクトが，SCO なら，SCO は，今終わった学習セッション中に，学習者とのやり取りに関して，より詳細な情報を通信していた可能性がある．ランタイムデータモデルを通して SCO によって通信された，後続のシーケンシング評価に影響を与える情報に責任を持つのは LMS の役割である．ランタイムデータモデルセクションは，シーケンシング評価に影響するデータモデル要素を SCO に関連付けられた学習アクティビティにマッピングする LMS の役割を記述している．シーケンシング評価でランタイムデータの適切な時点での利用を確実にするために，LMS が各データイベントのランタイムデータ変化に対応することが推奨される（「セクション 3 Application Programming Interface」参照）．

いくつかのケースで，コンテンツ作成者は，SCO が終了したあとで（SCO のコンテクストで「終わった」がどのような意味をしようとも），利用者と SCO のやり取りを許したくないことがある．このようなケースでは，SCO が起動したウインドウのタイプによって以下の動作が許される：

1. SCO が起動された画面がトップレベルの画面なら(画面は親画面を持たないが opener を持つなら)，SCO は，Terminate(“”)をコールしたあと画面を閉じることができる．SCO がこのように動作するという要件はない．LMS は，いつそのようなイベントが起こるか監視するために，SCO を起動した独立したポップアップ画面の状態をモニターすることが推奨される．これは，LMS が学習者に適切な実装で規定されたユーザーインターフェースを提示することを可能にする．

2. トップレベル画面でないなら(親画面をもつなら)，SCO は親画面もしくは親に繋がるどの画面にも影響を与えることはできない．例えば，SCO は，自身のウインドウでない限り，トップウインドウを閉じる事が許されない．

注意：このようなケースでは，推奨される動作は，LMS が取り除くまでの間，SCO が中立で受動的なコンテンツを表示することである．

セクション 3
アプリケーションプログラムインターフェース (API)

このページは空白である．

3.1. アプリケーションプログラミングインターフェース (API)

3.1.1. API 概要

SCORM バージョン 1.2 を含むそれまでの初期のバージョンの SCORM は，AICC's CMI001 相互運用性のためのガイドライン [7]で定義されたランタイム環境機能に基づいていた．それ以後，AICC は，相互運用性のためのガイドラインの各部分を標準化することに力点をおいて，彼らの仕事を IEEE Learning Technology Standards Committee (LTSC)へ提出した．SCORM は，IEEE 1484.11.2-2003 学習技術標準-ECMAScript Application Programming Interface for Content to Runtime Services Communication [2] を用いるとしている．この規格は，コンテンツとランタイムサービス(RTS：run-time service)の通信のための API を記述している．RTS は，学習コンテンツの実行と配信をコントロールするソフトウェアと定義され，リソース割り当て，スケジューリング，入出力制御およびデータ管理などのサービスを提供する [2]．SCORM の観点からは，用語「RTS」および「LMS」は同義の用語である．API は，一般的には ECMAScript [9] (JavaScript としてより広く知られている) を利用した API サービスの共通のセットにより，コンテンツと LMS によって提供される RTS との間のデータ通信を可能にする．このセクションでは，IEEE 規格で使われている用語「コンテンツ」は，SCO に対応する．(なぜなら SCORM では，SCO は API を利用して LMS に通信するコンテンツオブジェクトであるため)．

共通 API の使用は，相互運用性および再利用に対する SCORM のハイレベルな要求の多くを実現している．共通 API は，SCO が LMS と通信する標準化された手段を提供しているが，さらに SCO 開発者から特定の通信実装を隠ぺいしている．LMS が提供する API インスタンスが，どのように LMS のサーバ側の部分と交信するかについては SCORM の対象範囲外である．この対象範囲外の通信は，LMS ベンダの好きなやり方で実装することができる．

SCORM を通して利用されるいくつかの用語がある：API，API 実装，API インスタンスである．図 3.1.1a は，これらの用語とお互いの関係を表している．

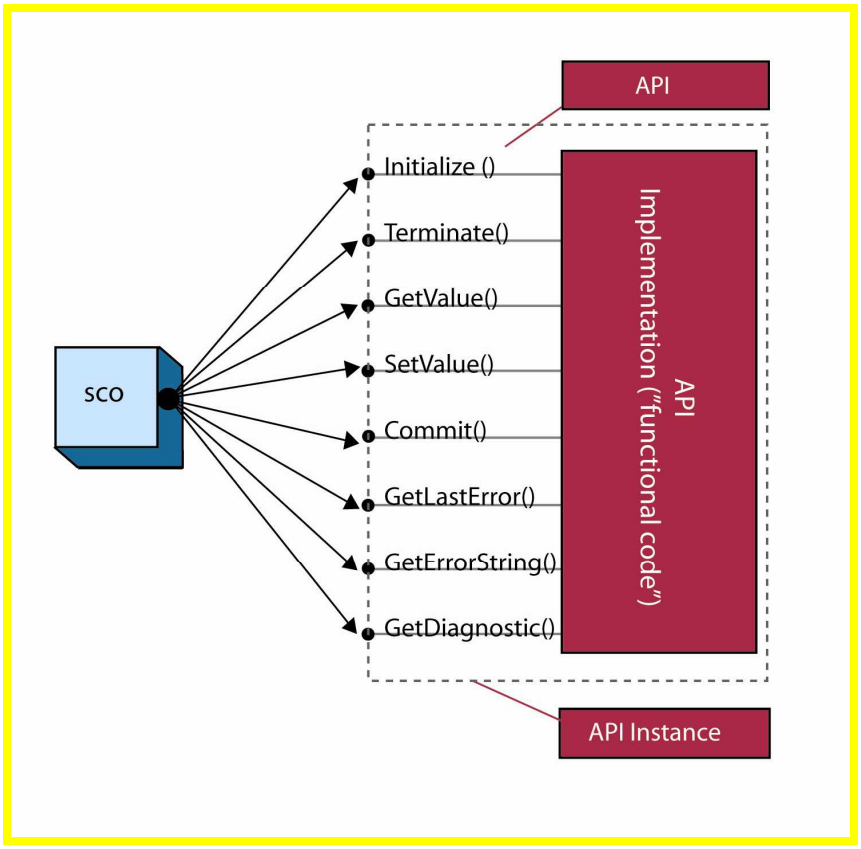

図 3.1.1a: API, API インスタンス, API 実装

簡単に言うと，API は，SCO が利用可能となったときに，依存することのできる一式の定義された関数群である．

API 実装は，API の機能を実行，提示する機能ソフトウェアの一部である．API 実装が，同一の公開インターフェースを利用し，そしてインターフェースのセマンティックに従う限り，API 実装がどのように機能するかは，SCO 開発者には関係ない．LMS に要求されるのは，API の機能を実装した API 実装を提供し，その公開インターフェースをクライアントの SCO に提示することだけである．

API インスタンスは，API 実装の個別の実行コンテクストおよび状態である[2]．API インスタンスは SCO のオペレーション中に SCO がやりとりする実行ソフトウェアの一部を意味する．

API の重要な側面は，SCO が LMS と通信する事を可能にするメカニズムを提供することである．SCO が一旦起動されると，情報を LMS に格納，および，から読出しできることが前提となる．全ての LMS と SCO 間の通信は，SCO によって開始される．SCO に実装された機能を LMS が呼び出すメカニズムは現時点では存在しない．

API 実装によって提示されるメソッドは，3 つのカテゴリーに分類される．以下の表 (表 3.1.1b) はこれらのカテゴリーを定義する．

表: 3.1.1b: API 手法

メソッド	説明
セッションメソッド	セッションメソッドは, API インスタンスを通した SCO と LMS 間の 通信セッションの開始と終了を示すのに利用される
データ転送メソッド	データ転送メソッドは, API インスタンスを通した SCO と LMS 間でデータモデル値を交換するのに利用される
サポートメソッド	サポートメソッドは, API インスタンスを通した SCO と LMS 間の補助的な通信 (エラー処理など)に利用される

3.1.2. APIメソッドおよびシンタックス

共通 API の使用は, 相互運用性および再利用に対する SCORM のハイレベルな要求多くを実現している. 共通 API は, SCO が LMS と通信するための標準化された手段を提供しているが, SCO 開発者から特定の通信実装を隠ぺいしている. このことは, SCO が一貫した方法で API インスタンスを見つけることができるなら, 成立する. そうでなければ, コンテンツ開発者は, 彼らのコンテンツを異なった LMS ベンダーのシステム上で機能するように適応させなければならない. これが, DOM 階層中で, LMS が API インスタンスを提供する位置に制約があり, API インスタンスに検索するための共通の名称を付与する, 主な理由の一つである.

API を取り扱うとき守るべきいくつかの一般的な要件がある:

- 全ての関数名は大文字と小文字を区別し, その通りに表示されなければならない
- 全ての関数パラメータすなわち引数は大文字と小文字を区別する.
- 支援メソッドを除き, すべての API 関数の呼び出しはエラーコードを設定する.
- パラメータと返り値は全て文字列で引き渡され, API 通信を用いるデータモデルで記述されたデータ型およびフォーマットと互換でなくてはならない. 整数, 実数と時刻の全ての値は ECMAScript の文字列へのキャスト変換で変換されるようにエンコードされる. SCORM のいくつかの例では, クオート(“”)記号のあるリターンデータが含まれていることに注意せよ. この記号は返ってきた文字列の一部ではない. この記号は, 値が文字列であることを示すために使用されている.

API の重要な側面は, SCO が LMS と通信する事を, API が可能にすることである. SCO が, 一旦起動されると, 情報を LMS と交換(すなわち, “get” および “set”)することが前提となっている. LMS と SCO 間の全ての通信は, SCO によって開始される. 言い換えると, 通信の開始は SCO から LMS への一方向である. SCO は, 常に, LMS の API インスタンスの機能を呼び出す. LMS は, SCO に定義されたいかなる機能も呼び出さない. これは, API インスタンスが値を返す概念と混乱してはいけない. それは, SCO が開始した呼び出しに対する応答としてのみ行われる. SCO によって実装された機能を LMS が呼び出すメカニズムは現時点では存在しない.

全ての API 関数は, 以下のセクションで詳細に記述される. いくつかの関数はデータモデルを参照する. データモデルは, セクション 4:SCORM ランタイム環境データモデルで詳細に記述される. エラー処理およびエラーコードは, セクション 3.1.7:API 実行エラーコードで詳細に記述される.

3.1.3. セッションメソッド

SCO は自身と API インスタンス間のデータ通信を開始および終了するのにセッションメソッドを使用する.

3.1.3.1 Initialize

メソッドのシンタックス: return_value = Initialize(parameter)

説明: この関数は，通信セッションを開始するのに利用される．LMS に固有の初期化項目の処理をおこなわさせる．

パラメータ: (“”) – 空の文字列． 空の文字列はパラメータとして引き渡さなくてはならない．

リターン値: この関数は二つのうち一つの値を返す．リターン値は，文字列で表現される．(“”)記号は返却される文字列 の一部ではなく，単に返される値を示すために用いられる．

- “true” –通信セッションの初期化が LMS で定義された通りに成功すると，文字列 “true”が返却される．
- “false” –通信セッションの初期化が LMS で定義された通りに成功しないと文字列 “false” が返却される．API インスタンスは，発生したエラー固有のエラーコードを設定する．SCO は，エラーの種類を特定するのに GetLastError()を呼び出すことができる．LMS は，エラーに関するより詳細な情報を GetDiagnostic()関数により，提供する場合がある．

3.1.3.2 Terminate

メソッドのシンタックス: return_value = Terminate(parameter)

説明: この関数は，通信セッションを終了するのに使用する．SCO が LMS とこれ以上通信する必要がないと決めたときこの関数を使用する．Terminate()関数は，最後に成功した Initialize(“”) もしくは Commit(“”)のどちらか最新の呼び出し以降に SCO に設定された全てのデータの保持を実行する(暗黙の Commit(“”)呼び出し)．これで，SCO が設定した全てのデータが LMS に保持されることが保証される．

一旦交信セッションが終了に成功すると，SCO はサポートメソッドをコールすることだけが許される

パラメータ: (“”) – 空の文字列．空の文字列はパラメータとして引き渡さなくてはならない．

リターン値: このメソッドは二つのうち一つの値を返す．リターン値は，文字列で表現される．(“”)記号は返却される文字列 の一部ではなく j，単に返される値を示すために用いられる．

- “true” –通信セッションの終了が LMS で定義された通りに成功すると，文字列 “true”が返却される．
- “false” –通信セッションの終了が LMS で定義された通りに成功しないと文字列 “false” が返却される．API インスタンスは，発生したエラー固有のエラーコードを設定する．SCO は，エラーの種類を特定するのに GetLastError()を呼び出すことができる．LMS は，エラーに関するより詳細な情報を GetDiagnostic()関数により，提供する場合がある．

3.1.4. データ転送メソッド

SCO は，通信セッション内で利用するデータの格納，読出しを指示するのにデータ変換メソッドを利用する．SCO は，ランタイムデータを LMS から/へ転送するのにこれらのメソッドを利用する．例えば，LMS は，アクティビティの完了/達成度を決定し，およびシーケンシングおよびナビゲーションの判断にこのデータを使用する．

3.1.4.1 GetValue

メソッドのシンタックス: return_value = GetValue(parameter)

説明: この関数は LMS から情報を要求する．SCO は LMS からの情報を要求し，以下のようなことを決定することができる：

- LMS がサポートするデータモデル要素の値
- LMS がサポートされるデータモデルのバージョン
- 特定のデータモデル要素がサポートされているか否か

パラメータ: パラメータは，データモデル要素を完全に指定する情報を表す

リターン値: このメソッドは二つのうち一つの値を返す．リターン値は，文字列で表示される．

- パラメータと関連する値を含む文字列
- エラーが発生すると，API インスタンスは，定められたエラーコードを設定し，空の文字列 (“”)を返す．SCO は，エラーの種類を特定するのに GetLastError()をコールすることができる．LMS は，エラーに関するより詳細な情報を GetDiagnostic()関数により，提供する場合がある．

SCO は，この関数から返された空の文字列を有効な値とみなすべきではない．SCO は，エラーコードをチェックし，エラーがなかったか確認すべきである．エラーがなければ，空の文字列は LMS から返された有効な値である．要求の処理中にエラー条件が発生したら，この状況は適切なエラーコードによって表示される (セクション 3.1.7: *API 実行エラーコード*参照)．

3.1.4.2 SetValue

メソッドのシンタックス: return_value = SetValue(parameter_1, parameter_2)

説明: このメソッドは，parameter_1 で定義されたデータ要素に対して parameter_2 の値を LMS へ転送するように要求するために用いられる．このメソッドは，SCO が LMS へ格納する情報を送信ことを可能にする．API インスタンスは，設定されたデータを直ぐに (サーバサイドコンポーネントに) 保持するか，もしくは，一旦ローカル (クライアントサイド) キャッシュに保存するように設計される．

パラメータ:

- parameter_1 –設定するデータモデル要素を完全に指定する情報
- parameter_2 – parameter_1 の内容に設定される値．parameter_2 の値は，parameter_1 で指定されたデータモデル要素に対して定義されたデータタイプに変換できる文字列である．

リターン値: このメソッドは二つのうち一つの値を返す．リターン値は，文字列で表示される．クオート(“”)記号は文字列 の一部として返されず，単に返される値を示すために用いている．

- “true” –LMS が parameter_1 の値を設定するのに parameter_2 の内容を受け入れれば，文字列 “true”が返される
- “false” – LMS が parameter_1 の内容を parameter_2 の値で設定しようとしてエラーが発生すれば，文字列 “false”が返される．SCO は，エラーの種類を特定するのに GetLastError()をコールすることができる．LMS は，エラーに関するより詳細な情報を GetDiagnostic()関数により，提供する場合がある．

3.1.4.3 Commit

メソッドのシンタックス: return_value = Commit(parameter)

説明: このメソッドは，どちらか直近に発生した最後の Initialize(“”) もしくは Commit(“”)呼び出し以降に，API インスタンスによってキャッシュされた可能性のある SCO からの全てのデータを，恒久的なデータ格納場所に転送することを要求する．LMS はエラーコードを“0” (発生したエラーなし)とし，“true”を返す．

API インスタンスが値をキャッシュしないのであれば，Commit(“”)は “true”を返し，エラーコードを“0” (発生したエラーなし)に設定し，他の処理を実行しない．

キャッシュされたデータは，commit データメソッド呼び出しによって変更されてはならない．例えば，SCO がデータモデル要素の値を設定し，commit データメソッドを呼び出し，その後，同じデータモデル要素の値を取得すれば，返された値は commit データメソッドを呼び出す前に設定された値でなくてはならない．Commit(“”)メソッドは，SCO によって予防メカニズムとして使用することができる．このメソッドは，SetValue()に設定されたデータが，通信セッションが妨害されたり，異常に終了したり，もしくは Terminate(“”)が呼ばれる前に早まって終了してデータが失われる可能性を減らすことを保証するために使用することができる．

パラメータ: (“”) –空の文字列．空の文字列をパラメータとして渡さなくてはならない．

リターン値: このメソッドは二つのうち一つの値を返す．リターン値は，文字列で表される．クオート(“”)記号は文字列 の一部として返されず，単に返される値を示すために用いている．

- “true” –データが長期格納場所に保持されることに成功すると，文字列 “true”が返される
- “false” –データが長期格納場所に保持されることに失敗すると，文字列 “false”が返される．API インスタンスは，発生したエラーに固有の値にエラーコードを設定する．SCO は，エラーの種類を特定するのに GetLastError()をコールすることができる．LMS は，エラーに関するより詳細な情報を GetDiagnostic()関数により，提供する場合がある．

3.1.5. サポートメソッド

サポートメソッドは，SCO がエラー処理および診断情報を決定することができるようにするためのものである．これまで説明してきた各 API 関数においてのみエラーは発生する．これらのエラーが発生するごとに，エラーコードは発生したエラーを示すために変更される．サポートメソッドの呼び出しは，エラー状態に影響を与えない．言い換えると，どんなサポートメソッドを呼び出しても，現在のエラーコードが変わる事はない．これらのサポートメソッドは，エラーが発生したかどうか，およびエラー条件にどのように対応するかを SCO が決定することを可能にする．

3.1.5.1 GetLastError

メソッドのシンタックス: return_value = GetLastError()

説明: このメソッドは，API インスタンスの現在のエラー状態に対するエラーコードを要求する．SCO がこのメソッドを呼び出した場合，API インスタンスは現在のエラー状態を修正してはならず，単純に要求された情報を返す．

セッションメソッドもしくはデータ転送メソッドが成功したか確認することが推奨される．現在のエラーコードを返すのに GetLastError()を用いることができる．関数の処理中にエラーが発生すると，SCO は問題を解決するのに適切な手順を踏む事ができる．

パラメータ: API メソッドはどんなパラメータも受け入れない

リターン値: APIインスタンスは，APIインスタンスの現在のエラー状態を反映するエラーコードを返す．返される値は最後に発生したエラーのエラーコードを表す文字列 (変換可能な 0 から 65536 までの整数)である．

3.1.5.2 GetErrorString

メソッドのシンタックス: return_value = GetErrorString(parameter)

説明: GetErrorString()関数は，現在のエラー状態のテキスト表記を検索するのに利用される．この関数は，SCO が，パラメータの値によって指定されるエラーコードのテキスト表記を要求するのに用いられる．APIインスタンスは，セクション 3.1.7: *API実行エラーコード*で指定されるエラーコードをサポートする役割がある．このコールは現在のエラー状態に影響はなく，単純に要求された情報を返すだけである

パラメータ:

- パラメータ: エラーメッセージに対応するエラーコードの文字列 (整数値)を表す

リターン値: このメソッドは，パラメータの値で指定されるエラーコードの表記からなるテキストメッセージを返す．全てのリターン値で以下の要件が遵守されなくてはならない：

- リターン値は最大長 255 文字の文字列である
- SCORM は，文字列のテキストが何を含むかについて，どんな要求もしない．エラーコード自身は，明示的および排他的に定義されている．エラーコードに対するテキスト表記は，LMS 固有である．
- 要求されたエラーコードが LMS にとって不明なら，空の文字列 (“”)が返される．空の文字列が返されるのはこの時だけである

3.1.5.3 GetDiagnostic

メソッドのシンタックス: return_value = GetDiagnostic(parameter)

説明: GetDiagnostic()関数は，LMS の特定の用途のために存在する．それは，LMS が API インスタンスを通して，追加の診断情報を定義する事を可能にする．このコールは現在のエラー状態に影響がなく，単に要求された情報を返すだけである

パラメータ:

- パラメータ: 実装者固有の診断に関する値．最長 255 文字．パラメータの値は，エラーコードでもよいがあるが，エラーコードに制限されているわけではない

リターン値: APIインスタンスは，診断情報を表す文字列を返す．返される文字列の最長は 255 文字である．LMS にとってパラメータが不明なら，空の文字列 (“”)が返される

注意：GetDiagnostic()関数に渡されるパラメータが空文字列 (“”)であれば，関数は最後に起こったエラーに関する診断情報を表す文字列を返すことが推奨される．

3.1.6. 通信セッション状態モデル

API インスタンスが存在している間，API インスタンスが遷移する概念的な状態モデルが IEEE によって定義されている．図 3.1.6a は，ある SCO に対して ランタイム時に API インスタンスが遷移する状態を表している．API インスタンスの状態は，特定のイベントへの API インスタンスの遷移を特定する．各々の定義された API インスタンスの状態により，SCO がどの関数を呼び出すことができるかが定義される．API インスタンスに起こる状態は，以下のように定義される：

- Not Initialized
- Running
- Terminated

実装において，状態モデルを実装することが要求されないことに注意せよ．状態モデルは，典型的な通信セッションにおける，API 関数の意図された動作を示すための概念的なモデルにすぎない．

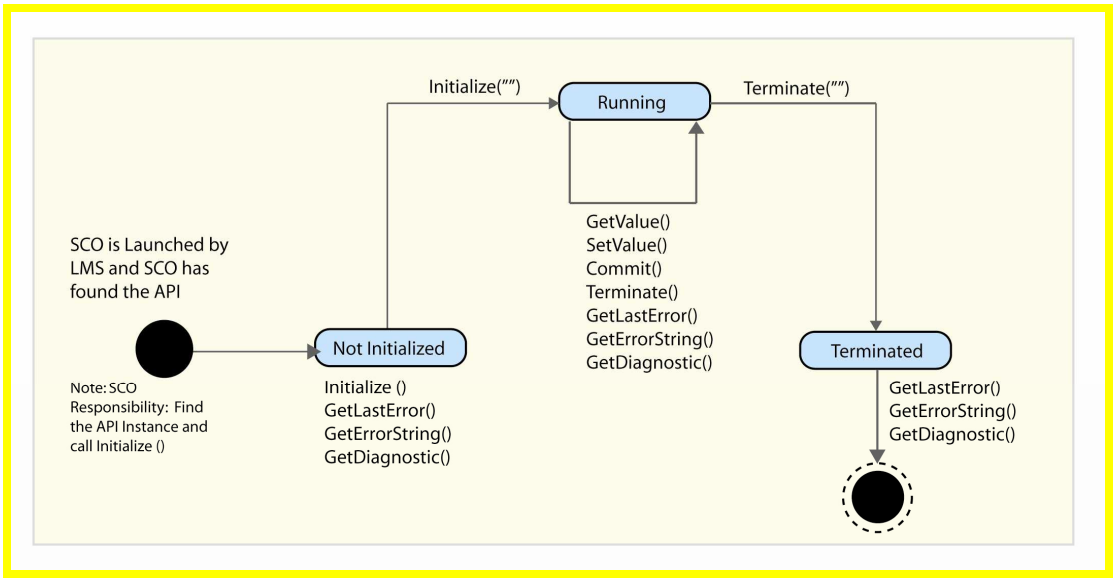

図 3.1.6a: 概念的な API インスタンス状態遷移

Not Initialized: これは，SCO の実際の起動と，SCO が Initialize("") API メソッド呼び出しに成功する間の概念的な通信状態を表す．この状態の間，LMS が提供する API インスタンスを見つけるのは SCO の役割である．SCO は以下の API 関数を用いる事が可能である：

- Initialize()
- GetLastError()
- GetErrorString()
- GetDiagnostic()

Running: これは，SCO が Initialize("") API メソッド呼び出しに成功し，Terminate("") API 呼び出しに成功する前の概念的な交信状態を表す．SCO は以下の API 機能を用いる事が可能である：

- Terminate()
- GetValue()
- SetValue()
- Commit()
- GetLastError()

- GetErrorString()
- GetDiagnostic()

Terminated: これは，SCO が Terminate("") API メソッド呼び出しに成功したときの概念的な通信状態を表す．SCO は以下の API 関数を用いる事が可能である：

- GetLastError()
- GetErrorString()
- GetDiagnostic()

3.1.7. API 実装エラーコード

全てのエラーコードは文字列として表現される整数であることが必要である．IEEE 規格[2]は，全てのエラーコードが 0 から 65536 までの間であることを要求している．この規格は，0 から 999 までの範囲を将来の版のために予約している．1000 から 65535 の間は，追加のエラーコードを実装やプロファイルによって定義する事ができる．SCORM は，エラー条件に使用される追加のエラーコードを定義することはない．これは，実装においてエラーコードを定義し，これらのエラーコードを実装で定義された方法で利用することを禁止するものではない．SCORM は，決められたエラーコードを，定義されたエラー条件で利用する事を要求するだけである(このセクションで定義される)．

サポートメソッド GetLastError(), GetErrorString()および GetDiagnostic()を除いた各 API 関数は，現時点で API インスタンスが保持するエラーコードを設定する．SCO は，直近の API 関数コールが成功したか，もし成功でなければ何が悪かったかを評価するのに，GetLastError()関数を呼び出すことがある．GetLastError()関数は，直近の API 関数コールによって起こったエラーの種類を決定するのに用いることのできるエラーコードを返す．

IEEE は以下のカテゴリーおよび様々なエラーコードに対する範囲を定義している：

表 3.1.7a: エラーコードカテゴリーおよび範囲

エラーコードカテゴリー	エラーコード範囲
エラーなし	0
一般的なエラー	100 – 199
シンタックスエラー	200 – 299
RTS エラー	300 – 399
データモデルエラー	400 – 499
実装で定義されたエラー	1000 – 65535

3.1.7.1 API 関数の呼び出しの成功

API 関数実行中にエラーが発生しなければ，LMS は API インスタンスのエラーコードを 0 に設定する．これは，要求の処理中に，API 関数の呼び出しでエラーが発生しなかったことを示す．

API 関数呼び出しの後にエラーコードが 0 であれば，実際に呼び出された関数ごとに以下を仮定できる：

- Initialize(“”): APIインスタンスが，LMS固有の適切な通信セッション初期化手順の実行に成功した．通信セッションが確立され，APIインスタンスは他の関数呼び出しに対する準備ができている．APIインスタンスの概念的通信状態は“Running”である．
- GetValue(parameter): 要求したデータモデル要素の値が返される．要求による値は，LMSによって信頼があり正確であるとみなされる．概念的通信状態は変化しない
- SetValue(parameter_1, parameter_2): SetValue()の parameter_2 として渡された値は，LMSによって正しく設定された(parameter_1 で表わされたデータモデル要素と関連する値として保存された)．SetValue()`parameter_1` で用いられたデータモデル要素に対する GetValue()の要求は，LMSに保存された値を返す．概念的通信状態は変化しない
- Commit(“”): Initialize(“”) もしくは最後の Commit(“”)メソッド呼び出し以降に（SetValue()メソッド呼び出しを用いて）設定された値は，永続的なデータ格納場所に正しく転送された．このメソッドは，SCOへの同じ学習者アテンプト内でのその後の学習者セッション中にデータが入手可能であることを保証する．概念的通信状態は変化しない
- GetLastError(), GetErrorString() and GetDiagnostic(): これらの APIメソッドは，APIインスタンスのエラーコードに影響しないし，エラーコードを変えることはない．
- Terminate(“”): APIインスタンスは，LMS固有の適切な通信セッション終了手順の実行に成功した．通信セッションは終了した．APIインスタンスの概念的通信状態は “Terminated”である．

3.1.7.2 一般的なエラーコード

一般的なエラーコードは，APIメソッド要求の処理中に発生したエラーを表す．これらのエラーはいくつかの条件に基づいて用いられる：

- 概念的通信状態モデルの現在の状態
- APIインスタンスによって処理されている APIリクエストの種類

3.1.7.2.1 一般的な例外 (101)

The General Exception error condition indicates that an exception occurred and no other specific error code exists. The API Instance shall use the General Exception error code in scenarios where a more specific error code is not defined.

一般的な例外エラー状態は，例外が発生し，他の固有のエラーコードが存在しないことを示す．APIインスタンスは，より具体的なエラーコードが定義されていない場合に，一般的な例外エラーコードを利用する

3.1.7.2.2 一般的な初期化失敗 (102)

一般的な初期化失敗エラー状態は，通信セッションを初期化しようとしているときに不具合が発生したことを示す．概念的通信状態が“Not Initialized”で，通信セッション初期化プロセスが失敗し，他の特定のエラーコードが存在しないときに，APIインスタンスは一般的な初期化失敗エラーコードを用いる．概念的通信状態は元の状態(“Not Initialized”)のままである．

3.1.7.2.3 初期化済み (103)

初期化済みエラー状態は，通信セッションが既に初期化に成功した後に，SCOが通信セッションの初期化を試みたことを示す． SCOが通信セッション中に一度以上通信セッションを初期化(Initialize(“”))しようと試みたときに，APIインスタンスは初期化済みエラーコードを用いる．このエラーコードは，概念的通信状態が“Running”で，通信セッションを初期化する要求がなされたときに用いられる．この場合，APIインスタンスはエラーコードを “103”に設定し “false”を SCOへ返す．概念的通信状態は，元の状態(“Running”)のままである．

3.1.7.2.4 コンテンツインスタンス終了済み (104)

コンテンツインスタンス終了エラー状態は，通信セッションが既に終了していることを示す．このエラー状態は，SCO が Terminate(“”)メソッドへの呼び出しが成功した後に Initialize(“”)メソッドを呼び出そうとするときに発生する．このエラーコードは，概念的通信状態が “Terminated” で通信セッションを初期化する要求がなされたときに用いられる．この場合，API インスタンスはエラーコードを “104”に設定し “false”を SCO へ返す．この概念的通信状態は元の状態(“Terminated”)のままである．

3.1.7.2.5 一般的な終了失敗 (111)

一般的な終了失敗状態は，通信セッションを終了しようとするときに不具合が起こったことを示す．概念的通信状態が“Running”で通信セッション終了プロセスが失敗し，他のエラー情報（より具体的な通信セッション終了エラー）がないときに API インスタンスは一般的な終了失敗エラーコードを用いる．API インスタンスはエラーコードを “111”に設定し “false”を SCO へ返す．この概念的通信状態は元の状態(“Running”)のままである．

3.1.7.2.6 初期化前の終了 (112)

初期化前の終了エラー状態は，SCO が通信セッションを初期化する前に（概念的通信状態は“Not Initialized”），通信セッションを終了しようとしていることを示す．SCO が Initialize(“”)への呼び出しを成功する前に Terminate(“”)を呼び出そうとするとき，API インスタンスは初期化前の終了のエラーコードを用いる．API インスタンスはエラーコードを “112”に設定し “false”を SCO へ返す．概念的通信状態は元の状態(“Not Initialized”)のままである．

3.1.7.2.7 終了後の終了 (113)

終了後の終了エラー状態は，通信セッションが既に正しく終了した後に，SCO が通信セッションを終了しようとしたことを示す．前回の Termination(“”)が既に処理に成功した後に，SCO が Termination(“”)メソッドを呼び出したときに，API インスタンスは終了後の終了のエラーコードを用いる．API インスタンスはエラーコードを “113”に設定し “false”を SCO へ返す．概念的通信状態は元の状態(“Terminated”)のままである．

3.1.7.2.8 初期化前のデータ取得 (122)

初期化前のデータ取得エラー状態は，通信セッションの初期化が成功する前に SCO がデータを取得しようとしたことを示す．Initialize(“”)メソッドへの呼び出しが成功する前に SCO が GetValue()メソッドを呼び出したときに，API インスタンスは初期化前のデータ取得エラーコードを用いる．API インスタンスはエラーコードを “122”に設定し空の文字列 (“”)を SCO へ返す．概念的通信状態は元の状態(“Not Initialized”)のままである．

3.1.7.2.9 終了後のデータ取得 (123)

終了後のデータ取得エラー状態は，通信セッションの終了が成功した後に，SCO がデータを抽出しようとしたことを示す．Terminate(“”)メソッドへのコールに成功した後に，SCO が GetValue()メソッドを呼び出したとき，API インスタンスは終了後のデータ取得エラーコードを用いる．API インスタンスはエラーコードを “123”に設定し，空の文字列 (“”)を SCO へ返す．概念的交信状態は元の状態(“Terminated”)のままである．

3.1.7.2.10 初期化前のデータ保存 (132)

初期化前のデータ保存エラー状態は，通信セッションの初期化が成功する前に，SCO がデータを保存しようとしたことを示す．Initialize(“”)メソッドへの呼び出しが成功する前に，SCO が SetValue()メソッドを呼び出したときに，API インスタンスは初期化前のデータ保存エラーコードを用いる．API インスタンスはエラーコードを “132”に設定し，“false”を SCO へ返す．概念的通信状態は元の状態(“Not Initialized”)のままである．

3.1.7.2.11 終了後のデータ保存 (133)

終了後のデータ保存エラー状態は，通信セッションの終了が成功した後に，SCO がデータを保存しようとしたことを示す．Terminate(“”)メソッドへのコールに成功した後に，SCO が SetValue()メソッドを呼び出したときに，API インスタンスは終了後のデータ保存エラーコードを用いる．API インスタンスはエラーコードを “133”に設定し，“false”を SCO へ返す．概念的通信状態は元の状態(“Terminated”)のままである．

3.1.7.2.12 初期化前のコミット (142)

初期化前のコミットエラー状態は，通信セッションの初期化が成功する前に，SCO がデータを永続的記憶にコミットしたことを示す．Initialize(“”)メソッドへの呼び出しに成功する前に，SCO が Commit(“”)メソッドを呼び出したとき，API インスタンスは初期化前のコミットエラーコードを用いる．API インスタンスはエラーコードを “142”に設定し，“false”を SCO へ返す．概念的通信状態は元の状態(“Not Initialized”)のままである．

3.1.7.2.13 終了後のコミット (143)

終了後のコミットエラー状態は，通信セッションの終了が成功した後に，SCO が永続的記憶にコミットしたことを示す．Terminate(“”)メソッドへのコールに成功した後に，SCO が Commit(“”)メソッドを呼び出したとき，API インスタンスは終了後のコミットのエラーコードを用いる．API インスタンスはエラーコードを “143”に設定し，“false”を SCO へ返す．概念的通信状態は元の状態(“Terminated”)のままである．

3.1.7.3 シンタックスエラーコード

シンタックスエラーコードは，API メソッドの文法に関係するエラー状態を表す．現時点で，IEEE 規格は，シンタックス固有エラー状態を取り扱う一つのエラーコードを定義してきている．以下のセクションは，定義されたエラー状態およびその利用シナリオを表す．

3.1.7.3.1 一般的な引数エラー (201)

一般的な引数エラー状態は，API 関数に無効な引数を渡す試みがなされ，他の定義されたエラー状態が適用できないことを示す．もし発生した場合，より具体的なエラー条件を特定するためにデータモデルエラーを使用すべきである．エラー状態を表すのに他のエラーコードが使用できなければ，API インスタンスはエラーコード“201”を使用すべきである．このエラーコードが使用されるひとつの場合は，パラメータが以下の API コールに渡されたときに発生する：

- Initialize(“”)
- Terminate(“”)
- Commit(“”)

これら 3 つの API コールの全ては，空の文字列パラメータが渡されるという制約がある．SCO がこれらの関数呼び出しに他の引数を渡すと，API インスタンスはは “false”を返し，エラーコードを “201”に設定する．概念的通信状態は元のまま変わらない．

3.1.7.4 RTS エラーコード

RTS エラーコードは，RTS の実装に関係したエラー状態を表す．現時点で，IEEE 規格は，ランタイムに固有のエラー条件を扱う三つのエラーコードを定義している．以下のセクションは，こららの定義されたエラー状態およびその利用シナリオを表す．

3.1.7.4.1 一般的な Get 失敗 (301)

一般的な Get 失敗エラーは，一般的な Get の失敗が発生し，エラーに関する他の情報（より具体的なエラーコード）が，入手できないことを示す．このエラー状態は，GetValue()リクエストの処理のあらゆるものに対応したエラー状態として作動する．データ取得イベント(GetValue())エラーが発生し，API インスタンスが報告するのにより具体的なエラー条件（より具体的なエラーコード）を決定できないときに，API インスタンスは一般的な Get 失敗のエラーコードを用いる．API インスタンスはエラーコードを “301”に設定し，空の文字列 (“”)を SCO へ返す．このエラー条件は，概念的通信条件が“Running”の場合のみ発生する．このエラー条件が発生すると，概念的通信状態は元の状態(“Running”)のままである．

3.1.7.4.2 一般的な Set 失敗 (351)

一般的な Set 失敗エラー状態は，一般的な Set 失敗が発生し，エラーに関する他の情報（より具体的なエラーコード）が入手できないないことを示す．このエラー状態は，SetValue()リクエストの処理のあらゆるものに対応したエラー状態として作動する．データ保存イベント(SetValue())エラーが発生し，API インスタンスが報告するのにより具体的なエラー条件（より具体的なエラーコード）を決定できないときに，API インスタンスは，一般的な Set 失敗のエラーコードを用いる．API インスタンスはエラーコードを “351”に設定し，“false”を SCO へ返す．このエラー条件は，概念的通信条件が“Running”の場合のみ発生する．このエラー条件が発生すると，概念的通信状態は元の状態(“Running”)のままである．

3.1.7.4.3 一般的な Commit 失敗 (391)

一般的な Commit 失敗エラー状態は，一般的な Commit 失敗が発生し，エラーに関する他の情報（より具体的なエラーコード）が入手できないことを示す．このエラー状態はあらゆるものに対応した状態として作動する． Commit データイベント(Commit(“”))エラーが発生し，API インスタンスが報告するのにより具体的なエラー条件（より具体的なエラーコード）を決定できないときに，API インスタンスは一般的な Commit 失敗のエラーコードを用いる．API インスタンスはエラーコードを “391”に設定し，“false”を SCO へ返す．このエラー条件は，概念的通信条件が“Running”の場合のみ発生する．このエラー条件が発生しても，概念的通信状態は元の状態(“Running”)のままである．

3.1.7.5 データモデルエラーコード

API の特徴の一つは，コンテンツがデータを LMS へ送信し LMS からデータを取得できることである．このようなイベントの処理中に，あるエラー状態が発生することがある．IEEE 規格は，一般的なデータモデルエラーを表すいくつかのエラー状態を定義している．以下のセクションは，こららの定義されたエラー状態およびその利用シナリオを記述する．

3.1.7.5.1 未定義データモデル要素 (401)

未定義データモデル要素エラー状態は以下を示す：

- GetValue(parameter)でパラメータとして渡されたデータモデル要素が未定義で API インスタンスが認識できない．この状態は，API インスタンスが認識できないデータモデル要素を使おうとする試みがなされたことを示す．

- SetValue(parameter_1, parameter_2)で parameter_1 として渡されたデータモデル要素が未定義で API インスタンスが認識できない. この状態は, API インスタンスが認識できないデータモデル要素を使おうとする試みがなされたことを示す.

認識できない, または, 未定義のデータモデル要素は, SCORM ランタイム環境データモデルによって正式に定義されていない要素, もしくは API インスタンスに認識されない実装に即して定義された拡張要素である. 上記の二つのうち一つのシナリオに遭遇したとき, API インスタンスは未定義データモデルエラーコードを用いる. API インスタンスは:

- GetValue()要求に対して: エラーコードを "401"に設定し空の文字列("")を返す. このエラー状態は, 概念的通信条件が, "Running"の場合のみ発生する. このエラー状態が発生した場合, 概念的通信状態は元の状態("Running")のままである.
- SetValue()要求に対して: エラーコードを "401"に設定し"false"を返す. このエラー状態は, 概念的通信状態が"Running"の場合のみ発生する. このエラー状態が発生した場合, 概念的通信状態は元の状態("Running")のままである.

3.1.7.5.2 非実装データモデル要素 (402)

非実装データモデル要素エラー状態は以下を示す:

- GetValue(parameter)のパラメータとして渡されたデータモデル要素は, API インスタンスが認識することができるが, 実装されていない.
- SetValue(parameter_1, parameter_2)の parameter_1 として渡されたデータモデル要素は, API インスタンスが認識することができるが, 実装されていない.

LMS は全ての SCORM ランタイム環境データモデル要素を実装することを要求している. このエラー条件は, SCORM ランタイム環境データモデル要素にアクセスするときには発生してはならないが, 拡張データモデル要素をアクセスするときに発生する可能性がある.

3.1.7.5.3 未初期化データモデル要素値 (403)

未初期化データモデル要素値エラー状態は, SCO が初期化されていないデータモデル値を取得しようとしたことを示す. データモデル要素はいくつかの方法で初期化される:

- LMS によって: いくつかのデータモデル要素は, コンテンツパッケージで定義された値によって初期化される. いくつかのデータモデル要素は, LMS に定義された学習者登録プロセスもしくは学習者プロファイリングの要求事項によって初期化される
- SCO によって: いくつかのデータモデル要素は SCO によって初期化される

SCO が初期値を持たない要素に対して GetValue()を呼び出そうとしたとき, API インスタンスは未初期化データモデル要素値エラーコードを持ちいる. API インスタンスは, エラーコードを "403"に設定し, 空文字列 ("")を返す. 空文字列 ("")値は, データモデル要素要求に対しする正当な値である場合がある. 従って, 返された値は信頼できない場合があり, SCO は値が信頼できるがどうかエラーコードを確認するべきである. エラーコードが "0" – No Error に設定されたら, これは要求されたデータモデル要素に対して, 空の文字列が現在の値として LMS に保存されていることを示す. このエラー状態は, 概念的通信条件が"Running"の場合のみ発生する. このエラー状態が発生すると, 概念的通信状態は元の状態("Running")のままである.

3.1.7.5.4 読み出し専用データモデル要素 (404)

読み出し専用データモデル要素エラー状態は，SCO が読み出し専用として実装された要素のデータモデル値を保存しようとしたことを示す．このエラー状態は，SCORM ランタイムデータモデル，SCORM ナビゲーションデータモデルもしくは SCORM 環境で利用される他のデータモデル（拡張データモデル）の使用において発生することがある．SCO が読み出し専用のデータモデル要素に対して SetValue()を呼び出そうとしたとき，API インスタンスは読み出し専用データモデル要素エラーコードを用いる．API インスタンスは，エラーコードを “404”に設定し，“false”を返す．このエラー状態は，概念的通信条件が “Running”の場合のみ発生する．このエラー状態が発生すると，概念的通信状態は元の状態 (“Running”)のままである．

3.1.7.5.5 書き込み専用データモデル要素 (405)

書き込み専用データモデル要素エラー状態は，SCO が Write-only として実装された要素のデータモデル値を取得しようとしたことを示す．このエラー状態は，SCORM ランタイムデータモデル，SCORM ナビゲーションデータモデルもしくは SCORM 環境で利用される他のデータモデル（拡張データモデル）の使用において発生することがある．SCO が書き込み専用のデータモデル要素に対して GetValue()を呼び出そうとしたとき，API インスタンスは書き込み専用データモデル要素エラーコードを用いる．API インスタンスは，エラーコードを “405”に設定し，空の文字列 (“”)を返す．このエラー状態は，概念的通信条件が “Running”の場合のみ発生する．このエラー状態が発生すると，概念的通信状態は元の状態 (“Running”)のままである．

3.1.7.5.6 データモデル要素タイプ不整合 (406)

データモデル要素タイプ不整合エラー状態は，SCO が，データモデル要素に対してデータモデル値を保存しようとし，その値が正しいデータ型でなかったことを示す．このエラー状態は，SCORM ランタイムデータモデル，SCORM ナビゲーションデータモデルもしくは SCORM 環境で利用される他のデータモデル（拡張データモデル）の使用において発生することがある．SetValue()の parameter_2 として渡された値が，SetValue()の parameter_1 で示されたデータモデル要素に対して，正しい型もしくは定義されたフォーマットでないと評価されたとき，API インスタンスはデータモデル要素タイプ不整合エラーコードを用いる．API インスタンスはエラーコードを “406”に設定し“false”を返す．このエラー条件は，概念的通信条件が “Running”の場合のみ発生する．このエラー状態が発生すると，概念的通信状態は元の状態 (“Running”)のままである．

3.1.7.5.7 範囲外データモデル要素値 (407)

範囲外データモデル要素値エラー状態は，SCO が，データモデル要素に対してデータモデル値を保存しようとし，その値がその要素に固有の値の範囲内でないことを示す．このエラー状態は，SCORM ランタイムデータモデル，SCORM ナビゲーションデータモデルもしくは SCORM 環境で利用される他のデータモデル（拡張データモデル）の使用において発生することがある．SetValue()の parameter_2 として渡された値が SetValue()の parameter_1 で示されたデータモデル要素に対する範囲外であるとき，API インスタンスは範囲外データモデル要素値エラーコードを用いる．API インスタンスは，エラーコードを “407”に設定し“false”を返す．このエラー条件は，概念的通信条件が“Running”の場合のみ発生する．このエラー条件が発生しても，概念的通信状態は元の状態(“Running”)のままである．

3.1.7.5.8 データモデル従属関係未確立 (408)

データモデル従属関係未確立エラー状態は，適切な従属関係が確立されていない場合に利用される．従属関係は，他のデータモデル要素の前に設定すべき一つ以上のカギとなるデータモデルの値を表す．SCORM ランタイム環境データモデルにおける依存関係データモデル要素はセクション 4：SCORM ランタイム環境データモデルで記述される．

このエラー条件は，IEEE 規格では GetValue()および SetValue()要求中に発生する状況に利用されると記述されているが，SCORM では GetValue()リクエスト処理中にこのエラーコードを用いる状況は記述しない．SetValue()リクエストに対して，いくつかのデータモデル要素は，ある特定のデータモデル要素を他のデータモデル要素の前に設定しなくてはならないいう要件を持つ．要素を決められた順序で設定する事により，従属の完全性が維持される．これらの従属性要件のいずれかが確立されないと，LMS は API インスタンスエラーコードを“408”に設定し，“false”を返す．このエラー条件は，概念的通信条件が“Running”の場合のみ発生する．このエラー状態が発生すると，概念的通信状態は元の状態(“Running”)のままである．

3.1.7.6 SCORM 拡張エラー条件

SCORM ランタイム環境データモデル要素およびデータモデル要素のバインディング(ドット表記)の性質のため，SCORM では，SCORM 環境で起こりうるエラーシナリオを網羅するのに拡張エラー状態が定義されている．IEEE では，API インスタンスが拡張エラーコードの設定を許可するメカニズムは定義されていない．以下のセクションでは，SCORM は一連のエラー状態を定義する．これらのエラー状態が発生すると，API インスタンスは以下のように動作する：

- エラーコードを “301” (GetValue()失敗に対し)もしくは“351” (SetValue() 失敗に対し)に設定し“false”を返す
- SCO が発生したエラーに関してもっと情報を返すように要求したら (GetDiagnostic())，LMS は後続のエラー状態を記述する情報を返すことが推奨される

3.1.7.6.1 子を持たないデータモデル要素

子を持たないデータモデル要素エラー状態は，SCO が，子を持たないモデルデータ要素（データモデル要素に関しては SCORM ランタイム環境データモデル参照）に対して，サポートされたデータモデル要素（子）のリストを取得しようとしたことを示す．GetValue()でパラメータとして渡された値が子を持たないデータモデル要素に対する子要素の要求であれば（子要求の処理に関しては SCORM ランタイム環境データモデル参照），API インスタンスは子を持たないデータモデル要素エラー状態を用いる．API インスタンスはエラーコードを “301”に設定し，空の文字列 (“”)を返す．このエラー状態は，概念的通信条件が“Running”の場合のみ発生する．このエラー条件が発生すると，概念的通信状態は元の状態(“Running”)のままである．

例：GetValue(“cmi.learner_name._children”);

この例では，API インスタンスはエラーコードを“301”に設定し，空の文字列 (“”)を返す．learner_name 要素が子要素と考えられ，子は持たない．SCO がエラーに関して (GetDiagnostic()を呼び出すことにより) 更に情報を要求すれば，LMS は「このデータモデル要素は子を持たない」のように，データモデル要素が子を持たないことを示す文字列を返すことが推奨される．

3.1.7.6.2 カウントを持たないデータモデル要素

カウントを持たないデータモデル要素エラー状態は，SCO が，配列でないデータモデル要素に対して，現在保存されたエントリーの数（カウント）を取得しようとした示す (配列であるデータモデル要素および配列の処理に関しては SCORM ランタイム環境データモデル参照)．GetValue()のパラメータとして渡された値が，アレイでないデータモデル要素に対する現在保存されたエントリーの数（カウント）の要求であれば，API インスタンスはカウントを持たないデータモデル要素エラー状態を用いる．API インスタンスはエラーコードを “301”に設定し，空の文字列 (“”)を返す．このエラー状態は，概念的通信条件が“Running”の場合のみ発生する．このエラー状態が発生すると，概念的通信状態は元の状態(“Running”)のままである．

例：

- GetValue(“cmi.learner_name._count”);

この例に対して，API インスタンスはエラーコードを“301”に設定し，空の文字列 (“”)を返す．learner_name 要素は集合体ではなく，従ってカウントを持たない．SCO が(GetDiagnostic()を呼び出すことにより)エラーに関して更に情報を要求すれば，LMS は，「このデータモデル要素は集合ではない．従ってカウントできない」のように，データモデル要素が配列ではなく数えられないことを示す文字列を返すことが推奨される

3.1.7.6.3 データモデル要素集合順序違反

データモデル要素集合順序違反状態は，インデックス番号(*n*)が配列中で次に利用可能な場所にないところへ SCO が値を設定しようとしたことを示す．集合 (セクション 4.1.1.3: *集合の処理*参照) は詰まった配列 (スキップされた場所がないこと) でなくてはならない．新たな値が配列に追加されるとき，それは次に利用可能な場所でなければならない．SCO はこの場所を_count keyword データモデル要素を用いて決定できる (セクション 4.1.1.5: *Keyword データモデル要素*参照)．

新たな配列へのエントリに対して SetValue()で parameter_1 として渡されたインデックス場所(*n*)に対する値が，配列で次に利用可能な場所でない場合に，API インスタンスはデータモデル要素集合順序違反状態を用いる．API インスタンスはエラーコードを“351”に設定し，“false”を返す．このエラー状態は，概念的通信条件が“Running”の場合のみ発生する．このエラー状態が発生すると，概念的通信状態は元の状態(“Running”)のままである．

例：

- SetValue(“cmi.objectives.0.id”,”identifier_1”);
- SetValue(“cmi.objectives.2.id”,”identifier_2”);

この例では，API インスタンスは，エラーコードを “351”に設定し，“false”を返す．2 番目の要求は，アレイ中で，学習目標 ID のポジション 2 を，ポジション 1 の前に設定しようとしている．SCO が (GetDiagnostic()を呼び出すことによって)エラーについて更に情報を要求すれば，LMS は，「データモデル要素集合の設定順序違反」のように，データモデル要素集合の設定順序が不適切であることを示す文字列を返すことが推奨される．

3.1.7.6.4 範囲外データモデル集合要素要求

範囲外データモデル集合要素要求エラー状態は，SCO が集合中のデータモデル要素を取得しようとし，提供されたインデックス値が現在 LMS に保持されているものより大きいことを示す．集合 (セクション 4.1.1.3: *集合の処理*参照) は詰まった配列 (スキップされた場所がないこと) でなくてはならない． SCO は，保存されたデータモデル要素値の現在の数を_count keyword データモデル要素を用いて決定できる (セクション 4.1.1.5: *Keyword データモデル要素*参照)．

現在のデータモデルの集合が 4 つの値を持つとすると， SCO がアクセス可能なインデックス場所は 0, 1, 2, および 3 である．SCO が 3 より大きいインデックス場所から値を取得しようとすると，API インスタンスはエラーコードを “301”に設定し，空の文字列 “”を返す．このエラー状態は，概念的通信条件が “Running”の場合のみ発生する．このエラー状態が発生すると，概念的通信状態は元の状態 (“Running”)のままである．

SCO が(GetDiagnostic()を呼び出すことによって)エラーについて更に情報を要求すれば，LMS は，「データモデル要素要求は，インデックス値範囲外エラーのため，処理に失敗した」のように，データモデル集合要素が範囲外で存在しないことを示す文字列を返すことが推奨される．

3.1.7.6.5 データモデル要素未指定

データモデル要未指定素エラー状態は，SCO が，習得もしくは保存しようとするデータモデル要素を指定せず，GetValue()もしくは SetValue() API メソッド呼び出しを行ったことを示す．両 API メソッド呼び出しでは，データモデル要素が存在していなくてはならない：

- GetValue(parameter_1)
- SetValue(parameter_1, parameter_2)

GetValue()要求の parameter_1 もしくは SetValue()要求の parameter_1 として渡された値が (empty 文字列 – “”など)指定されていないときに，API インスタンスは，データモデル要未指定素エラー状態を用いる．

- GetValue(“”): API インスタンスはエラーコードを “301” (一般的な Get 失敗) に設定し，空の文字列 (“”)を返す
- SetValue(“”,”3.4”): API インスタンスはエラーコードを“351” (一般的な Set 失敗) に設定し，“false”を返す

このエラー状態は，概念的通信条件が“Running”の場合のみ発生する．このエラー状態が発生すると，概念的通信状態は元の状態(“Running”)のままである．SCO が(GetDiagnostic()を呼び出すことによって)エラーについて更に情報を要求すれば，LMS は，「データモデル要素が指定されていない」などデータモデル要素が指定されていないことを示す文字列を返すことが推奨される．

3.1.7.6.6 一意識別子制約違反

SetValue()リクエストで parameter_2 として渡された値が，一意でなくてはならない識別子を設定するのに用いられたときに，API インスタンスは一意識別子制約違反エラー状態を用いる．API インスタンスはエラーコード“351”を設定し “false”を返す．このエラー状態は，概念的通信状態が”Running”の場合のみに起こる． このエラー状態になると，概念的通信状態は，元の状態のままである(“Running”)．

例：

- API Call 1: SetValue(“cmi.objectives.0.id”,”objective1”)
- API Call 2: SetValue(“cmi.objectives.1.id”,”objective2”)
- API Call 3: SetValue(“cmi.objectives.2.id”,”objective1”)

上記の例では，配列のポジション 2 に保存された学習目標データの識別子は，学習目標の識別子を objective1 に設定しようとしている．この識別子は，配列のポジション 0 に保存された学習目標データに用いられている．cmi.objectives.n.id は，一意の識別子でなくてはならないので，これは一般的な Set 失敗エラーを起こす． LMS は要求された要素を設定せず，“false”を返し，エラーコードを“351”に設定する．SCO が，(GetDiagnostic()を呼び出すことによって)エラーについて更に情報を要求すれば，LMS は「データモデル要素の値は既に利用され，一意でない」など一意識別子制約違反があることを示す文字列を返すことが推奨される

3.1.7.6.7 最低限保証最大値超過 SPM（Smallest Permitted Maximum）

最低限保証最大値 (SPM: Smallest Permitted Maximum)超過エラー状態は二つの状況で利用される：

- SetValue()リクエストの parameter_2 として渡された値が，SPM を持つデータモデル要素を設定するのに用いられ，parameter_2 の長さがデータモデル要素用の SPM を超える場合，もしくは
- 値が集合に置かれ，集合の SPM が超過する場合

これらの状態は必ずしもエラー条件を表すことにはならないが，実装により，これらの状況が異なって処理されることがある．上記の両方の状況で，API インスタンスは，エラーコードを“0”に設定し “true”を返す．このエラー状態は，概念的通信条件が“Running”の場合のみ発生する．このエラー状態が発生すると，概念的通信状態は元の状態(“Running”)のままである．

SCO が，(GetDiagnostic()を呼び出すことによって)エラーについて更に情報を要求すれば，LMS は，「データモデル要素に設定される値は SPM を超えている．この値は切り捨てられる」のように SPM が違反されたことを示す文字列を返すことが推奨される

定義された SPM は，実装される格納容量の要求を意味する．実装は，最低 SPM の格納容量をサポートする必要がある．実装は SPM 以上の格納容量をサポートすることを選択することも可能である．実装が SPM だけをサポートし，文字列の切り捨てを実行するなら，実装が GetDiagnostic()メソッドコールを通して文字列の切り捨てが発生したことを示す情報を提供することを ADL は推奨する

3.1.7.6.8 バージョン無しデータモデル要素

バージョン無しデータモデル要素エラー状態は，SCO が，データモデルのバージョン(_version)を，それが許されていないデータモデル要素に対して取得しようとしたことを示す．_version keyword は，SCORM ランタイム環境データモデルで特定されたベースデータモデル(cmi._version)にだけに適用されるべきである．GetValue()のパラメータとして渡された値が cmi._version 以外 (例えば cmi.learner_id._version)であるときに，API インスタンスはバージョン無しデータモデル要素エラー状態を用いる．API インスタンスは，エラーコードを“301”に設定し，空の文字列 (“”)を返す．このエラー条件は，概念的通信条件が“Running”の場合のみ発生する．このエラー条件が発生すると，概念的通信状態は元の状態(“Running”)のままである．

(GetDiagnostic()を呼び出すことによって)SCO がエラーについて更に情報を要求すれば，LMS は，「_version keyword が不正に利用された」のように，バージョン要求が無効であることを示す文字列を返すことが推奨される

3.1.7.6.9 識別子複数回設定

識別子複数回設定エラー状態は，SetValue()リクエストの parameter_2 として渡された値が設定済の値と異なる識別子を設定しようとした時に，API インスタンスによって使われる．API インスタンスはエラーコードを 351 に設定し，"false"を返す．このエラー状態は概念上の通信状態が"Running"の時のみ発生する．この状態が起こった時に，概念的な通信状態は変わらず"Running"のままとなる．

例:

- API Call 1: SetValue(〝cmi.objectives.0.id〞,〞objective1)
- API Call 2: SetValue(〝cmi.objectives.0.id〞,〞objective2)

上記の例では，配列上の 0 番の位置に格納された学習目標データの識別子が objective1 に初期設定されている．API 呼び出し 2 は学習目標の識別子を objective2 に設定しようと試みている．objective2 の値は既に設定されているものと異なるため，この処理は「一般的な Set 失敗」エラーを引き起こす．LMS は要求された要素を設定せずに"false"を返し，エラーコードを 351 に設定する．SCO が(GetDiagnostic()の呼び出しで)エラーに関する更なる情報を求めた場合，LMS は"The data model element's value is

already set and cannot be changed"のような, 識別子の一意性の制約が犯されたことを示す文字列を返すことが推奨される.

3.2. LMS の責任範囲

SCORM は, LMS が IEEE 規格および SCORM で定義されたように API のインスタンスを提供することを要求する. API インスタンスは, SCO に対して特定の実装の詳細を隠ぺいする. SCORM は, API インスタンスの基になる通信インフラに対していかなる制約も課さない. 以下のセクションでは, ここまで述べられていない, LMS 実装に関する API インスタンスの追加要件を述べる.

3.2.1. API Instance　API インスタンス

SCORM は, ここまでに述べた API の必要な機能を実装する API インスタンスを, LMS が提供することを要求する. SCO が LMS によって開発された API インスタンスを利用するために, どこでどのように API インスタンスに対するアクセスを提供するかに関して, LMS にはある要件がある. API インスタンスを見つける相互運用可能な手段を提供するために, LMS の API インスタンスは “API_1484_11”と呼ばれるオブジェクトとして DOM [8]を経由してアクセス可能でなければならない. LMS は, SCO に ECMAScript 経由で API インスタンスへアクセスする関数を提供しなければならない.

SCO が, LMS が提供する API インスタンスを見つけるために, LMS はある特定の DOM 階層内で SCO を起動しなくてはならない. LMS は, API インスタンスを含む LMS 画面の子画面もしくは子フレームのブラウザ画面で SCO を起動しなくてはならない.

図 3.2.1a は, LMS が DOM 階層内で API インスタンスを置く事が許されている有効な位置を表す. .IEEE 規格で定義されるアルゴリズムは, API インスタンスを特定の位置で特定の順序に探すように構成されている. LMS が API インスタンスを提供する固有の位置は, LMS の実装要件に基づいて決定される.

親チェーン: この場合, SCO およびアセットは, API インスタンスが HTML Frameset 内にある構造で起動される. この場合, LMS は, フレームセット内の親フレームの階層のどこかに API インスタンスを提供する. IEEE 規格で定義されたアルゴリズムは, API インスタンスが見つかるか, 親フレームがないことが分かるまで, 親チェーンを動いて API を探す.

オープナー: この場合, SCO およびアセットは, 新しい画面で起動される. この新しい画面は, しばしばポップアップ画面と呼ばれる. 新しい画面は SCO もしくはアセットを含む. この場合, LMS は API インスタンスをオープナー画面(SCO もしくはアセットが起動される新しい画面を起動もしくは開く役割をする画面)で提供できる. IEEE 規格で定義されたアルゴリズムは, SCO がオープナーで起動されたかどうか決定する. そうであれば, オープナーで API インスタンスを探す.

オープナーの親チェーン: この場合, SCO およびアセットは, 新しい画面で起動される. LMS がオープナー画面の親フレームに API インスタンスをおく. IEEE 規格で定義されたアルゴリズムは SCO がオープナーで起動されたかどうか決定する. そうであれば, オープナーで API インスタンスを探す. API インスタンスがオープナー画面で見つからなければ, アルゴリズムはオープナー画面が親を持つかどうか決定する. アルゴリズムは API インスタンスが見つかるか, 親フレームがないことが分かるまで, 親画面階層を探すように設計されている.

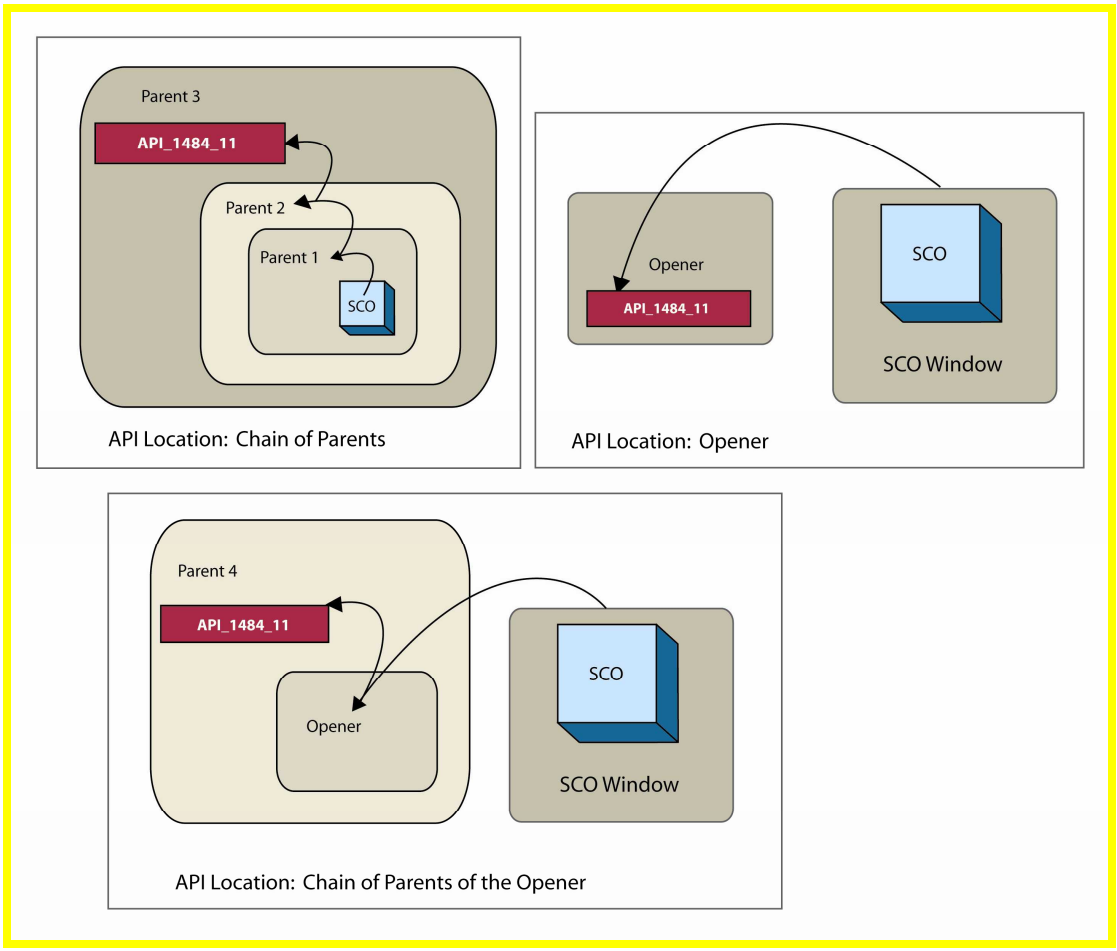

図3.2.1a: API 実行の許されたロケーション

LMS ベンダが SCO に API インスタンスへのアクセスを提供する代替メソッド（例えば Web サービスとして）の研究開発も進められている．しかし，SCORM は上記のメソッドしかサポートしない．DOM および ECMAScript は実績のある信頼できるテクノロジーで簡単に利用できる．SCORM の将来のバージョンは，IEEE 規格に定義された基本的要求をサポートする他の通信プロトコルを導入するかもしれない．

3.2.1.1 バージョン属性

IEEE 規格は，API インスタンスが DOM Object を経由してアクセス可能であることを要求する．この規格は，DOM オブジェクトがバージョンという属性を持つことも要求する．属性の値は API インスタンスのバージョンを表す．属性に割り振られる値は，メジャーおよびマイナーリリース値を整数として含むピリオドで仕切られた文字列からなる．API インスタンスが IEEE 規格に準拠して作られたことを示すために，最初の 3 個のキャラクターは“1.0”であることを要求されている．実装で，内部のバージョン情報を表すのに，値に追加のキャラクターを付け加えたい場合，4 番目のキャラクターはピリオドになる（例：1.0.）．

3.3. SCO の責任範囲

全ての SCO は，API を用いた通信をするときに特定の責任がある．SCO は，API インスタンスを一貫した方法で見つけることができなければならない．これは，DOM 階層内で LMS が API を提供する位置に関する制約，および，API インスタンスを見つけるための共通名称の存在に関する主な理由の一つである．API インスタンスが，DOM 階層のどこにでも存在して良いのなら，一貫性のある通信メカニズムを提供しランタイム環境を管理するのは極めて難しくなる．

3.3.1. API インスタンスの発見

SCO が LMS を用いて，学習者経験の追跡を始めるには，SCO は LMS が提供する API インスタンスを見つけなければならない．SCORM 環境では，コンテンツオブジェクトは Web ブラウザーで起動されるので，Web ブラウザーは API インスタンスを置く DOM を提供する．DOM は，あるページのオブジェクトの定義された構造もしくは組織とみなす事ができる．SCO がある LMS や他の LMS で API インスタンスを見つけるために，IEEE 規格は，この階層内のどこに API インスタンスを置くことができるか制約をかけている．重要なことは，SCO は指定された順序で以下のロケーションにおいて API インスタンスを探さなければならないことである：

1. (もし存在すれば) 現在の画面の親チェーンを，親チェーンの画面の先頭に到るまで
2. (もし存在すれば) オープナー画面
3. (もし存在すれば) オープナー画面の親チェーンを，親チェーンの画面の先頭に到るまで

SCO は API インスタンスをこの手順で探し，API インスタンスが見つかり次第停止しなければならない．SCO が探す対象を知るために，IEEE 規格は，API 実装に関連付けられた DOM のオブジェクトに対する必須名称を定義した．定義された名称は API_1484_11 である．

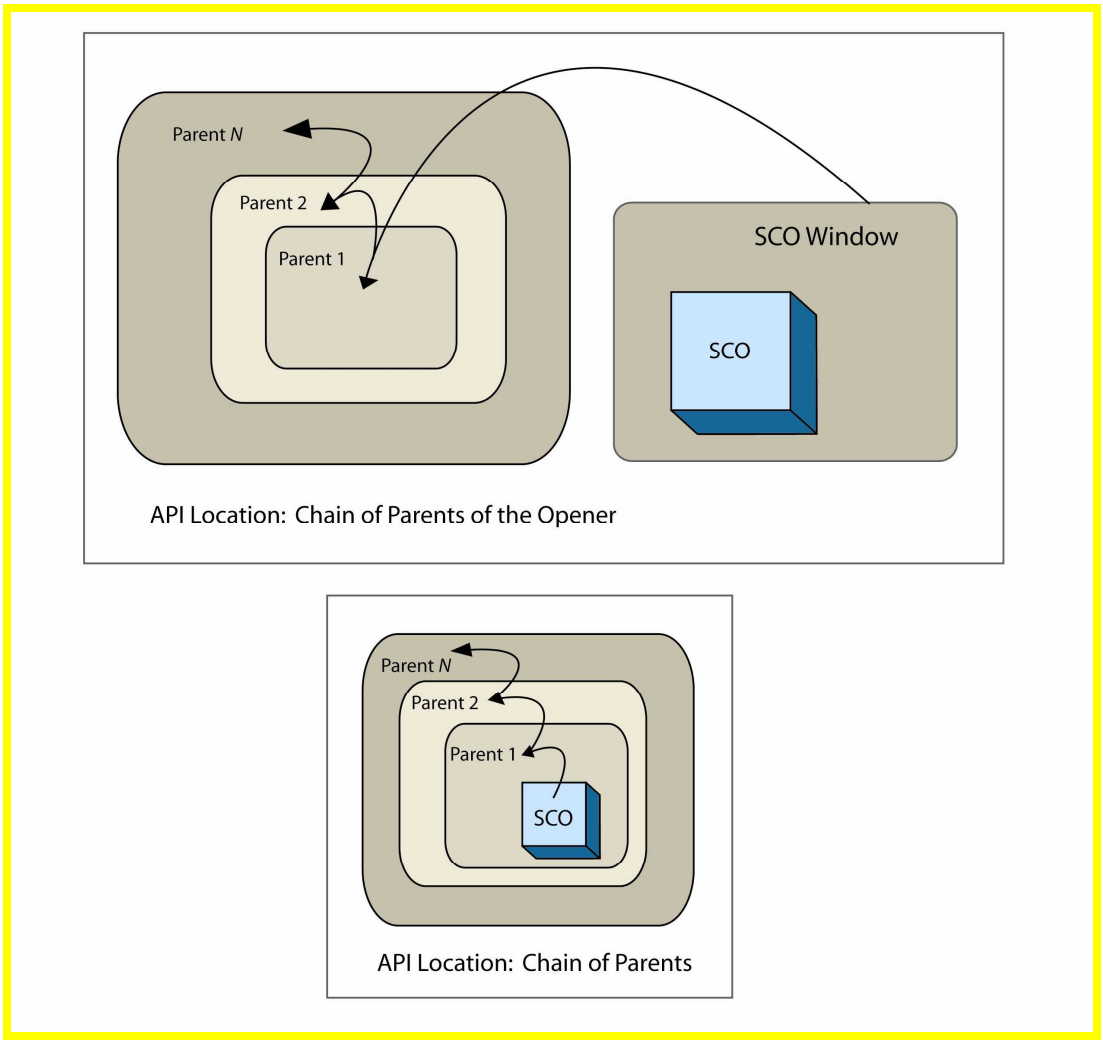

図 3.3.1a: API 発見のイラストレーション

一旦 SCO が API インスタンスを見つけると，最低限 SCO は Initialize(“”)および Terminate(“”) API 呼び出しを出さなくてはならない．

IEEE 規格は，API インスタンスを一貫性のある方法で見つける簡単な ECMAScript を提供している．この ECMAScript コードを利用することは必須ではない．他のバリエーションを書くことも可能である．

```
var nFindAPITries = 0;
var API = null;
var maxTries = 500;
var APIVersion = "";

// The ScanForAPI()関数はAPI_1484_11と呼ばれるオブジェクトを関数に渡された画面で探す.
// オブジェクトが見つかれば，オブジェクトへの参照が呼び出し元関数に返される.
// インスタンスが見つかればSCOはLMSが提供するAPIインスタンスへハンドルを持つ
// この関数は現在の画面の親の最大数を探す.
// オブジェクトが見つからなければ
// 関数はnullを返す．この関数は親の数に基づいてwinパラメータへ渡された値を再割り当てする
// 関数コールの最後にwin variableが
// 親チェーンのトップ階層の親に設定される
function ScanForAPI(win)
{
   while ((win.API_1484_11 == null) && (win.parent != null)
           && (win.parent != win))
   {
      nFindAPITries++;
      if (nFindAPITries > maxTries)
      {
         return null;
      }
      win = win.parent;
   }
   return win.API_1484_11;
}

// GetAPI()関数はLMSが提供するAPIインスタンスを探すプロセスを開始する
// 関数は現在の画面を表すparameterを取得する
// 関数は特定の順序で探すように作られており
// LMSが提供するAPIインスタンスが見つかると停止する
// もし現在の画面が親を持てば.
// 関数は現在の画面の親を探すことによって開始する，APIインスタンスが見つからなければ,
// 次に関数はオープナー画面がないか確認する．もし
// 画面がオープナーを持てば，関数はオープナー画面でAPIインスタンスを探し始める.
function GetAPI(win)
{
   if ((win.parent != null) && (win.parent != win))
   {
      API = ScanForAPI(win.parent);
   }
   if ((API == null) && (win.opener != null))
   {
      API = ScanForAPI(win.opener);
   }
   if (API != null)
      {
         APIVersion = API.version;
      }
   }
}
```

コードイラストレーション3.3.1b: API実行を発見するECMAScriptのサンプル

このECMAScriptコードサンプルは，IEEE規格で定義されたアルゴリズムに基づいている．アルゴリズムは更新され，機能を表す説明テキストが追加され，不必要なエラーメッセージが削除され，アルゴリズムにあるwin variableをより文脈に沿うように説明するように更新された．IEEEもしくはSCORMで定義

された ECMAScript を利用することは必須ではなく，独自の ECMAScript を自由に作成することができる．このアルゴリズムは API インスタンスを，IEEE および SCORM で定義されたように探さなければならない．

3.3.2. API 利用要件およびガイドライン

このセクションは，API を用いて情報を LMS に通信するためのいくつかの追加要件およびガイドラインの概略である．

3.3.2.1 不測のイベント

SCO は，様々な形で構築されている．SCO を開発するとき，コンテンツ編集者は SCO のデザイン，どのように SCO が LMS で配信されるように意図されているか，および，学習者が SCO と対話する様々な方法に注意する必要がある．

例えば，いくつかの SCO はページの集合として，あるページから他のページへの内部 SCO ナビゲーションが可能なように構築されている．この設計では，コンテンツ編集者は SCO が Initialize(“”)コールを最初のページで呼び出し Terminate(“”)を最後のページだけに呼び出すように実装するかもしれない．では学習経験中に不測の動作が起こったらどうするか？不測のイベントはいくつかのカテゴリーに分類される：

- 意図しない終了
- 故意のユーザーアクション
- 不慮の終了 (コネクションの切断，ブラウザークラッシュなど)

いくつかの LMS は，Initialize(“”)を呼び出すことに成功した SCO が，Terminate()関数を呼び出す前に予想外にアクセスできなくなったことを検出したら，Terminate(“”)関数の働きをシミュレーションして，これらの異なったシナリオに対応する．これは，問題の原因（事故だったのか，故意のユーザーアクションなのか，単にできの悪い SCO なのか？）を推測する手立てがないために行われる．勿論，突発的な事故（通信切断，完全なブラウザークラッシュなど）があれば，クライアント側のキャッシュから Commit(“”)を通してコピーされたデータだけがサーバー側のデータベースに生き残ることになる．

なぜ Commit(“”)が API インスタンスにあるかという主な理由の一つは，各データ要素値の更新が，ネットワークを通してクライアントとサーバ間でリアルタイムで送信されるときに発生する通信遅延を最小化するためである．API の実装は，Commit(“”)が呼び出されたときに，データ状態を保持し送信するだけのクライアント側のキャッシュを自由に提供することができる．

SCO は，API インスタンスがクライアント側のキャッシュを提供しているか，もしくは，全ての更新データをサーバに送っているかを知る術がない．発生であろう不測の動作および問題を最小限にするために，SCO 開発者は以下の推奨事項を遵守すべきである：

- 記録を要する重要な事象が発生したときには，後で何が起こるかに関わらず，Commit(“”)を呼び出す
- すべての SetValue()呼び出しの後に Commit(“”) を呼び出さない – これはいくつかの LMS でパフォーマンス強化を無効にし重大な問題につながりかねない．まとまった SetValue()の呼び出しの後にのみ Commit(“”)を呼び出す
- Terminate(“”)の直前に Commit(“”)を呼び出すことは一般的に何の利益もない．Terminate(“”)コールの間，何も Commit する新しいものがないので被害は与えないが．

- SCO が学習者から取り去らわれる（ブラウザからアンロードされる）前に Terminate(“”)を呼び出す．SCO は，Terminate(“”)を不必要に再び呼び出すことがないように，呼び出しに成功したか記録を取っておくべきである．適切に開発された LMS はこれを処置でき，二回目のコールを無視してエラーステータスを適切に設定する．クライアントブラウザが Microsoft Internet Explorer なら，onunload ハンドラーではなく onbeforeunload ハンドラー で Terminate(“”) を呼び出す．この方が，SCO を取り去ることに関わる後続の処理の全てが整然と進む可能性が高くなる．他のブラウザーは，onbeforeunload と呼ばれるイベントを起こさない．従って onunload イベントに頼っている．Onunload イベント以外のどこかで通信セッションを終了するメカニズムを提供するのが賢明である．

- Terminate(“”)直前まで多量のデータをためておくのではなく，SetValue()および Commit(“”)を通信セッション中に随時用いる．何故なら，あるブラウザでは，onunload が実行されているにもかかわらず，onunload が起動したいくつかの処理，特にネットワーク経由のデータ転送処理の実行が終わる前に，次ページのローディングを開始する実例がある．これはバックエンドに大変な状況をもたらす．いくつかの実装では，データの一部は正確に commit されないことがある．unload が発生する前にデータがセーブされ commit していれば，このデータは恐らく安全である．

セクション 4
SCORM ランタイム環境データモデル

このページは空白である．

4.1. データモデルの概要

共通データモデルを設定する目的は，SCO に関して定義された情報を，異なる LMS 環境で追跡できるようにする事である．例えば，学習者の得点を追跡することが一般的な要件だと決定したとすると，コンテンツが LMS 環境に得点を通知する共通的な方法を設定する必要がある．SCO が独自の得点表記を用いると，LMS はその情報をどのように受け取り，保存し，処理するか分からない可能性がある．

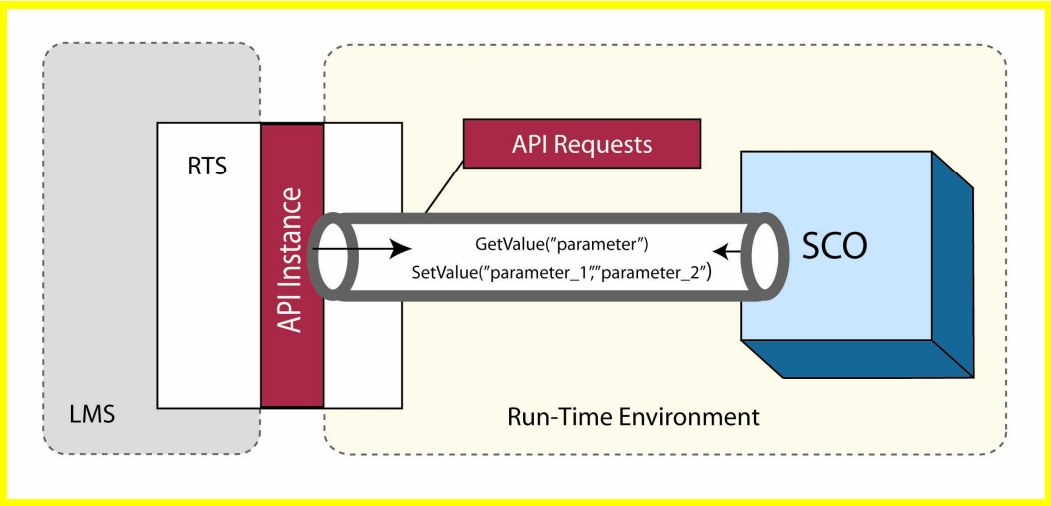

図 4.1.1a: API とデータモデルの利用

SCORM ランタイム環境データモデルは，IEEE LTSC CMI において作成された 1484.11.1 Standard for Learning Technology - Data Model for Content Object Communication [1]規格に基づいている．1484.11.1 は，コンテンツオブジェクト（すなわち SCORM における SCO）から LMS へ情報を交信するのに用いられるデータモデル要素の集合を定義する規格である．この情報には，学習者，SCO と学習者のやりとり，学習目標情報，成功ステータス，完了ステータスなどが含まれる．この情報は，多くの目的に対して，不可欠である．このデータは，学習者の進捗状況およびステータスを追跡し，シーケンシング決定を助け，学習者と SCO のやりとりを報告することなどに利用される．

このセクションのデータモデルは，SCORM ランタイム環境データモデルとして定義される．SCORM 2004 の前に，SCORM ランタイム環境データモデルは，AICC CMI001 Guideline for Interoperability [7]に基づいていた． SCORM バージョン 1.2 のリリースから，AICC は CMI001 を IEEE へ標準化のために提供した．SCORM 2004 は，IEEE 1484.11.1 Standard for Learning Technology Data Model for Content Object Communication [1]で定義されたデータモデルへの変更を導入する．IEEE 規格は純粋にデータモデル要素およびデータタイプを定義するので，SCORM は利用，動作，API との関係に則した要件を適用する必要がある．SCORM ランタイム環境データモデルは，特定のバインディング (dot-notation)，実装ガイダンスおよび IEEE 1484.11.1 規格の動作要件を定義する．

4.1.1. SCORM ランタイム環境データモデルの基礎

4.1.1.1 データモデル要素

データモデルを特定するのに，SCORM ランタイム環境データモデルで表わされるデータモデル要素の全ての名称は“cmi”で始まる．これは，これらのデータ要素が，IEEE 1484.11.1 Data Model for Content Object Communication draft standard[1]の一部であることを LMS に伝える．代替データモデルが開発されると，異なった識別子で始まることが予想される(cmi.*elementName* の代わりに adl.*elementName* 等々)もしくは dot-notation 以外の異なるバインディングを持つかもしれないことを意図している．

LMS は，SCORM で記述される全てのデータモデル要素を実装し，動作をサポートすることが要求される．

SCO によるいずれのデータモデル要素の使用もオプションである．SCO は API 機能 Initialize(“”)および Terminate(“”)を使用することだけを要求される． SetValue()もしくは GetValue()を使用することは要求されない．SCO は非常に小さく，詳細を追跡するように設計されていないことがありうる．しかし，SCO を追跡する場合は，複数の LMS 環境での再利用性のために，共通データモデルに準拠しなければならない．

全てのデータモデル要素の名称は dot-notation (cmi.success_status など)を使った ECMAScript 文字列にバインドされている．SetValue()メソッドコールでは，データモデル要素を設定するのに利用される全ての値は ECMAScript 文字列にバインドされている．ECMAScript 規格[9]は Unicode 標準 [13]をサポートし，これに準拠している．SCO および LMS は，これらの文字列は Unicode でコードされているので，Unicode エスケープシーケンスを含むことがあることに注意する必要がある．ブラウザで表示されるデータを取り扱うとき，SCO は異なったブラウザおよびブラウザバージョンでの Unicode に対するサポートレベルを知る必要がある．

4.1.1.2 シーケンシングへのデータモデルの影響

SCORM シーケンシング(SCORM SNbook 参照) は，定義されたシーケンシング情報，シーケンシング動作および学習者のコンテンツオブジェクトとの相互作用の結果に基づいて，コンテンツオブジェクトがどのように配信のために特定されるかを記述する．アセットは，シーケンシングに対して限定的な影響を持つ．LMS は，アセットが起動された事実だけを追跡する．一旦アセットが起動されると，アセットは完了したとみなされる．SCO は，学習セッション中，SCO と学習者のやりとりの結果を報告する事により，シーケンシングに影響を与えることができる．これは，SCO のランタイム環境データモデルを通して行われる．LMS は，SCO が SCORM ランタイム環境データモデルを通して報告した情報を用いて，後続の学習アクティビティのシーケンシングを制御しなくてはならない．SCORM は， SCO のランタイムデータがいつどのようにシーケンシングに用いられるかに対して特定の方法もしくはタイミングを強要しない．SCORM は，シーケンシング評価が必要とするときに，直近の情報が用いられることだけを規定する．このデータマッピングに対する具体的な LMS 要件は，各データモデル要素に分解された以下の表で提供される (より詳細情報は，*Sequencing Impacts* セクションの該当する表を参照されたい)．

例えば，SCO が学習者の SCO の完了を cmi.completion_status を通して報告すると，その SCO によって特定されるアクティビティも完了したと仮定される．

ADL Note: 現在経験中のコンテンツオブジェクトが SCO でありその上の学習者アテンプトが Abandon 又は Abandon All ナビゲーション要求によって終了するならば，データのマッピングは起こらない．破棄された学習者アテンプトは SCO と結びついたアクティビティの状態に影響を与えない．

4.1.1.3 集合の扱い

いくつかのデータモデル要素は，それらの要件に関連するデータを集める．データの集合は，この文書で，データのレコードとして参照される．各データのレコードは，配列の 1 要素として集められる．データのレコードは，配列におけるデータのレコードの位置を表すインデックス値によってアクセスされる．全ての配列は，インデックス値 0 から実装される (0 ベースアレイ)．

以下のデータモデル要素がデータのレコードの集合として定義される：

- 学習者からのコメント(cmi.comments_from_learner)
- LMS からのコメント (cmi.comments_from_lms)
- 学習目標(cmi.objectives)
- インタラクション (cmi.interactions)

これらのデータモデル要素は, SCO が複数のコメント, 学習目標および/もしくはインタラクションを追跡することができることを目的として存在する.

GetValue()を呼び出すとき発生し得る二つの明確なエラー条件がある:

1. 一般的な Get 失敗: 範囲外のデータモデル集合要素要求. データのレコードが要求された配列の位置に存在しない. 例えば, 学習目標に対してデータが配列の位置 0, 1 および 2 に存在し, SCO が学習目標情報に対して GetValue() 要求をポジション 4 に行うと, LMS はエラーコードを"301" (一般的な Get 失敗) に設定し, 空の文字列を返す. LMS が, GetDiagnostic() 要求に要求されたら, エラーに関してより具体的な情報 を SCO に提供することが推奨される. (セクション 3.1.7.6.4: *範囲外データモデル集合要素要求* 参照)

   起こりうるもう一つの展開は, SCO が新しいデータのレコードを配列の位置に作成しようとすることである.データモデル要素集合のいくつかは, 特定のデータモデル要素が最初に設定されることを要求する. そしてそれは LMS が適当な配列の位置でデータのレコードを設定することを許容する. これらのデータモデル要素が正しく設定されれば, 以下に従って集合のカウントを増やし新しいデータのレコードが集合に加えられることになる.

   これらのリストは, データの新しいレコードが作られる要因となるデータモデル要素のセットを特定する

   - cmi.objectives.n.id
   - cmi.interactions.n.id
   - cmi.interactions.n.objectives.m.id
   - cmi.comments_from_learner.n.comment
   - cmi.comments_from_learner.n.location
   - cmi.comments_from_learner.n.timestamp

   LMS がこれらのデータモデル要素に対する不正な SetValue() の要求を受けた時, LMS は false を返し適切な API のエラーコードを設定する. 保持する集合のサイズは増加しない.

   _count の要求が不正な SetValue()要求の前に行われ, 同じ _count 要求が不正な SetValue()要求の直後に起こった場合, 両方の呼び出しが返す値は同一となる(すなわち, 集合のサイズは変わらない).

   LMS がこれらのデータモデル要素の一つにおいて不正な SetValue()要求のすぐ後に GetValue()要求を受け取った時, 振舞いはこのようになる:

   (1) 空の文字列を返す

   (2) API エラーコードを 301(一般 Get 失敗)に設定する

2. 未初期化データモデル要素値. データのレコードが配列の位置に存在するが, 要求されたデータが値で初期化されていない. 例えば, SCO が学習目標の正規化された得点を取得する要求を出す (cmi.objectives.0.score.scaled), が, 正規化された得点はデータで初期化されていない. このケースでは, 学習目標のデータ記録はポジション 0 に存在する(cmi.objectives.0.id が設定されている), しかし, 正規化されたスコアは初期化されていない. LMS はエラーコードを "403" (未初期化データモデル要素値) を設定し, 空の文字列を返す.

SCORM ランタイム環境データモデルの現在の dot-notation バインディングのため，これらのデータ要素は配列の位置（インデックス）を表す整数値を含む．インデックスは単にインデックスである．インデックスは，ある SCO に固有とみなされるべきでない．LMS から LMS へ（もしくは学習セッションから学習セッションへ）同一学習目標が同一インデックスに保存されたという保証はないことを意味する．SCO 開発者は，これらのデータモデル要素を利用するときには，このことを念頭におくべきである．

学習目標とインタラクションのデータモデル要素は，各々の SCO の学習目標およびインタラクションに対して固有の識別子を示す識別子データモデル要素を含む．特定の学習目標もしくはインタラクションデータのインデックスを探すときにはこの値を用いなくてはならない．特定の学習目標もしくはインタラクションに対して値を検索するには，特定のインデックスポジションに頼るのではなく，ある識別子に対してインデックスを決定するために，全ての SCO の目標およびインタラクションのリストを探すことが推奨される．異なる学習セッションにおいて，インデックスポジションは同一とは限らない．全ての新しい集合要素は，順番に追加される．集合要素にアクセスするのに利用されるインデックスは，無意味な最初のゼロを持たない．“15”であり，“0015”ではない．値を集合に追加するとき，SCO はその集合の最後のインデックスポジションを特定しなければならない．SCO および LMS は，集合要素を作るもしくは直すときに，インデックスポジションをとばしてはいけない（パック配列）．_count キーワードデータモデル要素は，集合のデータモデル要素の現在の番号を決定するのに利用できる．例えば，SCO に対して現在保存されている学習目標の数を決定するには，以下の API コールが利用される：

```
var numOfObjectives = GetValue("cmi.objectives._count");
```

加えて cmi.comments_from_lms（このデータモデル要素は，SCO から見て read-only である）の全ての集合データモデル要素は，SCO に上書きされたサブ要素を持つことができる．上書きか追加かは SCO 開発者により，SCO の開発中に決められる事である．

集合中のデータモデル要素は，dot-number notation (.n で表される)を利用して参照される

cmi.objective.**n**.completion_status

例えば，SCO の最初の学習目標の完了ステータスデータモデル要素の値は，
"cmi.objective.0.completion_status"と表記され，4 番目の学習目標では
"cmi.objective.3.completion_status"と表記される．

4.1.1.4 SPM (Smallest Permitted Maximum)

SPM（Smallest Permitted Maxmum）が定義してデータモデル要素要求を表す場合には，二通りのケースある．これらの二つのケースは，文字列の長さに対してのケースと，集合（アレイ，バッグなど）に含まれるデータモデル要素の数のケースである．SPM は，実装が受け入れる或いは処理する，（集合の）エントリーの最小数もしくは，文字の最小数と定義される．実装は，SPM 以上に受け入れ，処理してよいが，少なくとも SPM はサポートしなければならない．例えば，

- 文字列: 文字列が SPM が 100 と定義されると，実装は最低 100 文字を受け入れてサポートしなければならない．実装は定義された SPM 以上をサポートすることも可能である．SCORM は，SPM 値以上の文字を含む文字列の扱いに関与しない (実装は文字列を切り捨てることも自由である)．実装は，文字列が定義された SPM 以上を含んだときの結果に気を付けるべきである．
- 集合: 集合(アレイ，セット，バッグなど)は，SPM が 100 と定義されると，実装は集合で最低 100 エントリーを認めてサポートしなければならない． 実装は定義された SPM 以上をサポートすることも可能である．SCORM は，SPM 値以上のエントリーを含む集合の扱いに関与しない (実装は新しいエントリーを受け入れないことも自由である)．実装はエントリーの数が定義された SPM をこえるときの結末を承知しておくべきである．

定義された SPM は，実装の格納領域要件を表す．上記の通り，実装は最低 SPM の格納領域をサポートしなければならない．実装は，SPM 以上の格納領域をサポートする事を選択できる．実装が，SPM だけをサポートし，文字列の切り捨てを実行するなら，実装が，文字列の打ち切りが発生したことを示す情報を GetDiagnostic()メソッドコールを通して提供することを ADL は推奨する

例えば，実装が(SPM の範囲だけをサポートするので)文字列を切り捨て，次に SCO が GetDiagnostic()を呼び出すと，実装は「SetValue()コールは成功した．値が SPM 以上であり打ち切られた」という文字列を返すことができる．

文字列と集合の全ての検証は値の実際の格納の前に行われるべきである．実装が SPM で文字列を切り捨てることを選択すると，検証プロセスは，切り捨ての前に文字列の検証を実施する．文字列に対する SPM は，コンテンツ開発者が利用するようになっている．コンテンツ開発者は，SPM および SPM を越えたとき何が起こるか注意する必要がある．コンテンツ開発者は最大の相互運用性のために SPM 制約を守ることが推奨される．

4.1.1.5 キーワードデータモデル要素

SCORM は，IEEE Content Object Communication Data Model のバインディングの中のキーワードデータモデル要素を定義する．3 つのキーワードデータモデル要素は_version，_count および _children である．キーワードデータモデル要素は，LMS が管理し，他のデータモデル要素の状態から導かれる．これらのキーワードは，SCO がデータモデル要素の説明情報を探す事を可能にする．キーワードデータモデル要素は，特定の他のモデル要素にのみ適用される（詳細情報はセクション 4.2 を参照）．キーワードデータモデル要素は read-only として実装される．SCO が，キーワードデータモデル要素を設定しようとすると（セクション 4.2 で定義されるように），LMS はエラーコードを“404”（データモデル要素は Read Only）に設定し “false”を返す．例えば，cmi.interactions._children._version は許されていない．この状況を SCO 試みると，LMS は空の文字列を返し，エラーコードを“401”（未定義のデータモデル要素）に設定する．

_version: _version キーワードデータモデル要素は，LMS にサポートされるデータモデル要素のバージョンを決定するのに利用される．このキーワードデータモデル要素は，どのデータモデル要素にも適用できない．（セクション 4.2.1: *データモデルバージョン*参照）

_count: _count キーワードデータモデル要素は，集合に現在あるデータモデル要素の数を決定するのに利用される．count は，集合にあるデータモデル要素の合計数である．この値は，情報を保存するために利用可能な次のインデックスポジションを決定するために，SCO によって要求されることがある．このキーワードデータモデル要素は，集合であるデータモデル要素にのみ適用できる．

_children: _children キーワードモデル要素は，LMS がサポートする，親データモデル要素中の全てのデータモデル要素を決定するのに利用される．有効な_children 要求から返された文字列は，LMS がサポートするすべてのデータモデル要素の，カンマで区切られたリストでなくてはならない．返されたデータモデル要素のリストは，セクション 4.2 に示されたデータモデル要素に相当する(大文字小文字を区別するデータモデル要素の名称が適用される)．データモデル要素名称は，予約されたトークンと考えられ，(リザーブされたトークンを区切るのに利用される)ブランクスペースが文字列で提供されたら，ブランクスペースは無視される．返されるデータモデル要素の文字列中の順番は関係ない．このキーワードモデル要素は子を持つデータモデル要素にのみ適用される．

4.1.1.6 予約されたデリミタ

dot-notation バインディングの性質により，いくつかのデータモデル要素は，文字列として簡単に表されるものより多くの情報を伝える必要がある．これらのケースでは，特殊な予約されたデリミタがバインディ

ングの要件を満たすために作成された．これらの特殊なデリミタは，SetValue()要求で用いられる文字列もしくは GetValue()要求から返される文字列に現れる．特殊な予約されたデリミタが必要とされるケースには次の場合がある：

- 特定の文字列に対して言語タイプを表す場合(Data Type: localized_string_type)
- インタラクションでの学習者の回答で順序が関係あるかないかを示す場合
- インタラクションでの学習者の回答で大文字小文字が関係あるかないかを示す場合
- 値をペアもしくはリストで表す場合

上記の各々のケースで，適用可能な場合デフォルト値が提供される．このデフォルト値は特殊な予約されたデリミタが指定されない場合に利用される．全てのケースで，予約されたデリミタは SPM の値に対してカウントされない．表 4.1.1.6a は，予約されたデリミタの詳細を表す．

表 4.1.1.6a: 予約された属性デリミタ

予約されたデリミタのシンタックス	デフォルト値	例
{lang=<language_type>}	{lang=en}	{lang=en}
{case_matters=<boolean>}	{case_matters=false}	{case_matters=true} {case_matters=false}
{order_matters=<boolean>}	{order_matters=true}	{order_matters=true} {order_matters=false}

これらのデリミターは必須でないため，デリミターが指定されないケースではデフォルト値が前提となる．デリミターが文字列で利用されると，デリミタの位置およびシンタックスに他の要件が発生する．

属性デリミタのシンタックス要件: デリミタは以下のフォーマットを持つ一定の文字の集合として扱われなくてはならない．

```
delimiter ::= "{" + name + "=" + value + "}"
```

"{"と"}"はデリミタの開始と終了の位置を示すために必要である．"="はデリミタを name と value に分割するために必要である．必要な文字が欠如すると，デリミタはシステムに認識されない．その代わりに，文字の集合は基になる文字列データの一部として扱われる．

name は デリミタの識別子を表す．name はリザーブされたトークンによって表される：

- lang
- case_matters
- order_matters

{lang=<language_type>}デリミタが文字列の言語情報として認識されるためには，それが適格な文字列の先頭に置かれなければならない．{lang=<language_type>}デリミタが文字列の先頭にない場合にはデフォルトの言語が仮定される．

name トークンは現状のまま(すなわち，以下のうちの 1 つ：lang, case_matters または order_matters)で記述されなければならない．これらのトークンのいかなる派生物(例えば，形だけのトークン名に空白を詰めている)もデリミタとして認識されず，文字の集合は基になる文字列の一部として扱われる．

value はデリミタの値を表す. デリミタの value 部分は以下のように制限されている:

- lang: language_type に表される値に制限される(language_type の要件に関してはセクション 4.1.1.7: *データタイプ*参照)
- case_matters: true もしくは false に制限される
- order_matters: true もしくは false に制限される

value がデリミタの型の要件を満たさないならばデリミタは不適切に作られ, 文字列も型の要件を満たさない(すなわち,『406 - データモデル要素タイプ不整合』のエラーとなる)

良い例:

- SetValue("cmi.comments_from_learner.0.comment",
  "{lang=en}Characterstring in the English language")

- SetValue("cmi.interactions.0.correct_response.0.pattern",
  "{lang=en}{case_matters=true}Characterstring in the English language where the case matters")

- SetValue("cmi.comments_from_learner.0.comment",
  "{lang =fr}Characterstring in the English language")

  この文字列を適格とするデリミタはなく, "{lang =fr}"は空白を含むため言語デリミタと見なされない. そしてそれは予約された言語デリミタと字句的に等価ではない. この例では文字列全体は{lang =fr}を含み, 依然として有効となる. デフォルトの言語は English ("en")である. この SetValue 呼び出しが行われると, LMS は"{lang =fr}Characterstring in the English language"の呼び出しに関連するデータモデル要素を設定し, エラーコードを 0 - No Error とし, true を返す.

- SetValue("cmi.comments_from_learner.0.comment",
  "{case_matters=invalid}Characterstring in the English language")

  この文字列を適格とするデリミタはなく, "{case_matters=invalid}"は cmi.comments_from_learner.n.comment として定義された形式の一部ではない. この例では文字列全体は{case_matters=invalid}を含み依然として有効, デフォルトの言語は English ("en")となる. この SetValue 呼び出しが行われると, LMS は"{case_matters=invalid}Characterstring in the English language"の呼び出しに関連するデータモデル要素を設定し, エラーコードを 0 - No Error とし, true を返す.

悪い例:

- SetValue("cmi.interaction.0.correct_response.0.pattern",
  "{case_matters=invalid}{lang=en}Characterstring in the English language")

  cmi.interaction.0 の型が記入済と仮定しても, 大文字小文字が関係するかを示すデリミタは true または false の値を持たねばならないため不適切となる. この例は"invalid"を使っている. デリミタが正しく形成されていないため, LMS は呼び出しに関連するデータモデル要素を設定しない. エラーコードを 406 - データモデル要素タイプ不整合 に設定し, false を返す.

- SetValue("cmi.comments_from_learner.0.comment",
  "{lang= fr}Characterstring in the French language")

  この例では, 言語デリミタが空白(空白は言語の文字列の一部とはならない)を含むため不適切である. デリミタが正しく形成されていないため, LMS は呼び出しに関連するデータモデル要素を設定しない. エラーコードを 406 - データモデル要素タイプ不整合 に設定し, false を返す.

要素デリミタの位置要件: デリミタは，文字列内で特定の位置に置かれるように要求されている．デリミタの組み合わせが用いられるケースでは，デリミタの順序はデータモデル要素によって記述される．デリミタの集合の中の一つのデリミタに対して（デリミタを明示ししないことにより）デフォルト値が利用されると，順序は依然維持される．デリミタは，デリミタ間にブランクスペースなしで結合しなくてはならない．例えば次のようなものである：

- {case_matters=true}{order_matters=true}

文字列中で最初に指定されるデリミターの前にブランクスペースもしくは他のキャラクターは許されない．デリミターがなければ（デフォルト値が利用されている事を示唆するが），値はデータモデル要素に用いられた文字列を表す．

空白やその他の文字が文字列の先頭に現れた場合，その文字の集合は基になる文字列の一部として扱われる．

上記のように，表 4.1.1.6a は 2 つの表に分けられている．表 4.1.1.6b は予約された分割デリミタを記述するために次のように更新される．

Table 4.1.1.6b: 予約された分割デリミタ

予約された分割デリミタ	デフォルト値	例
{lang=<language_type>}	{lang=en}	{lang=en}
{case_matters=<boolean>}	{case_matters=false}	{case_matters=true} {case_matters=false}
{order_matters=<boolean>}	{order_matters=true}	{order_matters=true} {order_matters=false}
予約された分割デリミタ	デフォルト値	例
[.]	なし，提供される必要あり	インタラクションに関連する値のペアを区分するのに用いられる 1[.]
[,]	なし，提供される必要あり	インタラクションの集合の値を区分するのに用いられる 1[.]a[,]2[.]c[,]3[.]b
[:]	なし，提供される必要あり	数値の領域のセパレータを表すのに利用される 数値が 1 から 100 の領域

分割デリミタのシンタックス要件: デリミタは以下のフォーマットを持つ一定の文字の集合として扱われなくてはならない．

```
delimiter ::= "[" + reserved_character + "]"
```

"["と"]"はデリミタの開始と終了の位置を示すために必要である. 必要な文字が欠如すると, デリミタはシステムに認識されない. その代わりに, 文字の集合は基になる文字列データの一部として扱われる.

reserved_character は定義済のセパレータを表す. reserved_character は一組の予約されたトークンによって表現される.

- .
- ,
- :

reserved_character トークンは現状のまま(すなわち, 以下のうちの 1 つ: [.], [,] または[:])で記述されなければならない. これらのトークンのいかなる派生物(例えば, reserved_character トークンに空白を詰めている)もデリミタとして認識されず, 文字の集合は基になる文字列の一部として扱われる.

分割デリミタの位置要件: デリミタは, 文字列内で特定の位置に置かれるように要求されている. デリミタは特定のデータモデル要素で定義される様々なデータの値を分割するために用いられる. これらの分割デリミタを用いるデータモデル要素について更なる情報を得たいなら, Section 4.2: SCORM Run-Time Environment Data Model を見ること.

4.1.1.7 *データタイプ*

各々のデータモデル要素は, 定義されたデータタイプを持つ. これらのデータタイプの全ての適用にあたっては, 定義された要件を遵守する事が求められている. このセクションは, データタイプの概略および各データタイプの具体的な要件を示す.

characterstring: Characterstring は ISO 10646 で定義された文字の列である. ISO 10646 は, Unicode Standard に相当するものである.

文字列型として定義される各々のデータモデル要素は LMS がサポートする SPM 値を指定する. これが特定の文字集合エンコーディング体系(例えば UTF-8, UTF-16)によって作られるオクテット文字列ではない文字の数, であることに注意する必要がある.

localized_string_type: ローカライズされた文字列は, 文字列の言語のインジケータを持つ文字列である. 言語情報が重要な特定のデータモデル要素がある. SCORM は, 文字列の言語を表すのに予約されたデリミターを適用する: {lang=*<language_type>*}.

文字列のフォーマットは以下のシンタックスを持つように要求されている:

"{lang=*<language_type>*}*<actual characterstring>*"

例:

"{lang=en}The content presented an excellent point dealing with the topic."

*{lang=<language_type>}*は, 後に続く文字列の言語を示すデリミタである. *{lang=<language_type>}*はオプションである. デフォルト language_type は, 指定されていない場合, "en" (英語)である. *<language_type>*値は, 250 文字の SPM で実装される.

言語タイプ: 言語を表すデータタイプ. このデータタイプのフォーマットは, 要求された言語コード(langcode), および, それに続く任意の複数のハイフン前置サブコード(subcode)からなる文字列である:

language_type ::= langcode ["-" subcode]*

langcode は:

- 2-文字コードは ISO 639-1 [14]によって定義される
- 3-文字コードは ISO 639-2 [15]によって定義される

- 値“i”は予約されており，Internet Assigned Numbers Authority (IANA)で定義される登録に対する接頭辞として利用される
- 値“x”は予約されており，プライベート利用に利用される

subcode は:

- 2-文字サブコードは ISO 3166-1 alpha-2 country codes[16]である
- 3 文字から 8 文字のサブコードは IANA で登録されている

追加的な subdode 値への要求は定義されない．langcode と subcode の値は 1 文字から 8 文字までの長さであること．

ISO 639-2 は，言語コードに対して二つのコードセットを特定する (目録 (bibliographic) および用語 (terminology))． どちらかのセットの利用も可能である．langcode および subcode は，大文字小文字の区別がある．SCORM では，language_type は以下のフォーマットで表すことを推奨する：

- langcode は通常小文字で提供される
- subcodes は通常大文字である

language_type は，特定のデータモデル要素に対して空の文字列であることが許されている．localized_string_type の language_type は，空の文字列であることを許されていない．localized_string_type の language_type 部分のデフォルトは “en” (英語)である．language_type は，このデータタイプとして定義された他のすべてのデータモデル要素で空の文字列の場合もある．(セクション 4.2.13: *学習者の好み参照*).

long_identifier_type: long_identifier_type は，ラベルもしくは識別子を表す．このラベルもしくは識別子は，SCO のコンテクスト内で固有である．long_identifier_type は，Universal Resource Identifiers (URI)のために定義されたシンタックスに準拠する文字列である (RFC 2396 [6]参照)． SCORM は，URI が，Uniform Resource Name (URN)に則ったグローバルに固有な識別子であることを推奨する (RFC 2141 [3]参照)．

全ての URN は以下のシンタックスを持つことを要求される：

<URN> ::= “urn:”<NID>“:”<NSS>

<NID>は Namespace Identifier および<NSS>は Namespace Specific String である[3].

例： urn:ADL:interaction-id-0001

空文字列が値を一意に識別させるだけの意味的情報を提供しないので，long_identifier_type 値は空文字列ではなく，また空白のみで占められることもない．long_identifier_type 値は，4000 キャラクターの SPM で実装される．

short_identifier_type: short_identifier_type はラベルもしくは識別子を表す．このラベルもしくは識別子は，SCO のコンテクスト内で固有である．short_identifier_type は，Universal Resource Identifiers (URI)に対して定義されたシンタックスに準拠する文字列である (RFC 3986 [6]参照)．

空文字列が値を一意に識別させるだけの意味的情報を提供しないので，short_identifier_type 値は空文字列ではなく，また空白のみで占められることもない．short_identifier_type 値は，250 キャラクターの SPM で実装される．

integer (整数) : 整数データタイプは，データモデル要素が正の整数(1,2,3,など)，負の整数 (-1, -2, -3 など)およびゼロ (0)であることを示す．

state (状態) : いくつかのデータモデル要素値は，定義された状態を持つ．これは，

```
state (browse,normal,review)
```

といった記述によって定義される．各状態は予約されたトークンに結び付けられている．これらの予約されたトークンはデータモデル要素で定義された状態値の文字列表現である．これらの予約されたトークンは，データモデル要素実装要件の Value Space セクションで定義されている．定義された各々の状態は必ずしも予約されたトークンに一対一でマッピングされない (not_attempted 状態値は，予約されたトークン not attempted に対応する)

real (10,7): real(10,7)データタイプは有効数字 7 桁の実数を表す

time (second, 10, 0): time (second,10,0)データタイプは，時間における一点を表す．このデータタイプは必須精度 1 秒およびオプション精度 0.01 秒を有する [1]．実装決定に影響を与えることがあるので，必須精度とオプション精度に気を付ける必要がある．例えば，アプリケーションがオプション精度を要求してもサポートされないていないことがある．SCORM の dot-notation バインディングは，時間における一点を表す文字列に対して特定のフォーマットを定義する．文字列のフォーマットは以下の通りである：

YYYY[-MM[-DD[Thh[:mm[:ss[.s[TZD]]]]]]]

- YYYY: 4 桁の年 (1970 >= YYYY >=2038)
- MM: 2 桁の月 (01 から 12 で 01=1 月)
- DD: 2 桁の日(01 から 31，年と月の値により異なる)
- hh: 2 桁の時間(00 through 23)
- mm: 2 桁の分(00 through 59)
- ss: 2 桁の秒(00 through 59)
- s: 1 桁もしくはそれ以上の 1 秒の小数点以下の桁数．小数点以下の桁数を用いる場合，SCORM では桁数は最大 2 桁と制約される (34.45 – 有効，34.45454545 – 無効)
- TZD: 時間帯指定子 (Time zone designator) (“Z” for UTC もしくは +hh:mm もしくは -hh:mm).
    - hh と mm は上記の hh と mm に準拠する．
    - 時間帯指定子は ISO 8601-2000 で定義されたように拡張フォーマットを利用する．拡張フォーマットは，時間と分のセパレータとしてコロン (:) の利用を要求する．ローカル時間と UTC に違いが要求されれば，時間は時間と分(03:10 など)で表現される，もしくは，時間差が時間だけであれば，時間のみ (03 など)で表現される．
    - 時間が 10 未満の場合，最初のゼロが要求される．
    - TZD 指定子が提供されないならば，デフォルトのタイムゾーンは「現地時間」と解釈される．
    - TZD が定義されるなら，hh:mm:ss.s も定義される(全てゼロかもしれないが)
- 最低 4 桁の年はなければならない．時間の他の部分が含まれたら，キャラクター文字“-“, ”T”, ”:” および“.” がキャラクター語彙表示の 1 部となる [1].

例：

- 2005
- 2005-07-25T03:00:00
- 2005-07-25T03:00:00-03:10
- 2005-07-25T03:30.35+05
- 2005-07-25T03:00:00Z

timeinterval (second, 10,2): timeinterval (second, 10, 2)は，データモデル要素 time interval に対する値は，0.01 の精度の経過時間であることを示す [1]．SCORM dot-notation バインディングは，時間インターバルを表す文字列に対するフォーマットを定義する．

文字列のフォーマットは，以下のとおりである．

P[yY][mM][dD][T[hH][mM][s[.s]S]]

- y: 年の数 (整数, >= 0, 制限なし)
- m: 月の数 (整数, >= 0, 制限なし)
- d: 日の数 (整数, >= 0, 制限なし)
- h: 時間の数(整数, >= 0, 制限なし)
- n: 分の数(整数, >= 0, 制限なし)
- s: 秒の数もしくは小数点以下の秒 (整数もしくは実数>= 0, 制限なし). 小数点以下の秒を用いる場合, SCORM は桁数を 2 桁に制限する (34.45 – 有効, 34.45454545 – 無効)
- 対応するゼロでない値がある場合, キャラクター文字指定子"P","Y","M","D","T","H","M","S"が現れる.

- ゼロ埋めされた値もサポートされる. ゼロ埋めは一組の文字集合で表される整数値を変化させない. 例えば, PT05H は PT5H や PT000005H と等価である.

例:

- P1Y3M2DT3H は, 1 年 3 ヶ月 2 日と 3 時間を示す
- PT3H5M は, 3 時間 5 分を示す

実装は, フォーマットとバインディングは SCO および LMS 間の交信のためであることに留意するべきである. フォーマットが期間を示すので, "PT5M"の期間は "PT300S"に等しい.

timeinterval 型のデータモデル要素が値を持つなら,

- 指定子 P は存在する;
- 年, 月, 日, 時, 分, または秒の値がゼロなら, 値と対応する文字リテラル指定は省略されるかもしれない. しかし少なくとも 1 つの文字リテラル指定と値は, 指定子 P に加えて存在する.
- 全ての時間構成要素(時, 分, そして秒)が使われないなら, 指定子 T は省略される. ゼロ値が, 時間構成要素 (例えば PT0S) のどれででも使われるかもしれない.

4.1.1.8 SCORM ランタイム環境データモデルの拡張

SCORM ランタイム環境データモデル自身が拡張されることはない. パラメータが cmi.*elementName* (*elementName* はセクション 4.2 で定義されていない他のトークン)である API 要求を LMS が受け取ると, LMS は以下の動作をする:

- GetValue(parameter): LMS は空の文字列を返し, エラーコードを "401"に設定する – 未定義データモデル要素
- SetValue(parameter_1, parameter_2): LMS は "false"を返しエラーコードを"401"に設定する – 未定義データモデル要素

他のデータモデル要素を定義し拡張する追加的な方法を取り上げる作業が現在起こっている. SCORM はこれらの進展を含むように更新されて行くことがある.

4.2. SCORM ランタイム環境データモデル

SCORM ランタイム環境データモデルは，SCO のランタイム時に LMS と SCO によって追跡されるデータモデル要素を含む．データモデル要素は，状態，得点，インタラクション，学習目標といった項目を追跡するのに利用される．いくつかのランタイム環境データモデル要素は，お互いに影響を与える，もしくは他と組み合わせて利用される．いくつかのデータモデル要素は，それらを用いると，同一コンテクスト（レッスン，コースなど）で用いられる他の SCO の制御と順序に影響を与える．データモデル要素の概要が表 4.2 に要約されており，その後，各データモデル要素が詳述されている．

表 4.2a: SCORM ランタイム環境データモデル要素の要約

データモデル要素	ドット表記バインディング	説明
Comments From Learner	cmi.comments_from_learner	学習者からのテキストからなる．
Comments From LMS	cmi.comments_from_lms	学習者が利用することを意図したコメントおよび注釈からなる．
Completion Status	cmi.completion_status	学習者が SCO を完了したかどうかを示す
Completion Threshold	cmi.completion_threshold	SCO が完了したとみなすかどうかを決めるために，学習者が SCO 完了に向けて進捗した度合いに対して比較を行う値を指定する．
Credit	cmi.credit	当該 SCO のパフォーマンスに関して学習者を評価するかどうかを示す
Entry	cmi.entry	学習者が SCO に以前アクセスしたかどうかを示す情報からなる
Exit	cmi.exit	学習者がどのようにあるいはどうして SCO を終了したかを示す
Interactions	cmi.interactions	測定もしくは評価の目的で，やりとりに関する情報を定義する
Launch Data	cmi.launch_data	SCO が初期化に利用する SCO 固有のデータを提供する
Learner Id	cmi.learner_id	SCO インスタンスが起動された学習者を指定する
Learner Name	cmi.learner_name	学習者の名前を表す
Learner Preference	cmi.learner_preference	学習者の SCO の使用に関連する学習者の好みを指定する
Location	cmi.location	SCO 中のロケーションを表す
Maximum Time Allowed	cmi.max_time_allowed	学習者アテンプトで SCO を利用するのに学習者が許された累積時間を示す
Mode	cmi.mode	学習者に SCO が提示されるモードを表す
Objectives	cmi.objectives	SCO と関連する学習目標もしくはパフォーマンス目標を指定する

Progress Measure	cmi.progress_measure	SCO の完了にむけて学習者が達成した進捗度合を指定する
Scaled Passing Score	cmi.scaled_passing_score	SCO に対する正規化された合格得点を指定する
Score	cmi.score	SCO に対する学習者の得点を指定する
Session Time	cmi.session_time	SCO に対する現在の学習セッションで学習者が費やした合計時間を指定する
Success Status	cmi.success_status	学習者が SCO を習得したかどうかを示す
Suspend Data	cmi.suspend_data	SCO への学習者アクセスもしくは学習者相互作用の結果，SCO が作成した情報を提供する
Time Limit Action	cmi.time_limit_action	既定最長時間を超えたときに SCO が何をすべきか示す
Total Time	cmi.total_time	現在の学習者セッションの前までの，現在の学習者試行で累積された全学習者の学習セッション時間の合計を指定する

以下のセクションでは，SCORM ランタイム環境データモデルの実装要件を定義する．各データモデル要素は新たな一つのセクション(4.2.1，4.2.2 など)で提示される．各セクションは，各データモデル要素に対する固有の要件を表す表を含む．これらの要件は LMS および SCO 実装両方に適用される．いくつかの要件は，LMS 実装に影響し，いくつかは SCO，いくつかは両方に影響する．

表 4.2b は，表のレイアウトとフォーマットを表す．以下の表は，表の各部分で記述される情報を提供する．

表 4.2b: データモデル要素表の説明

Dot-Notation バインディング	**詳細**
<データモデル要素の dot-notation 文字列表示>	このセクションはデータモデル要素のデータを提供する **データモデル要素実装要件:** 表のこの部分は，データモデル要素実装要件を定義する．この部分は，LMS および SCO 両方が遵守すべきデータタイプに関する要件を記述する．この部分は，データタイプ，値空間，フォーマットの 3 つのサブセクションに区分される． • **データタイプ**: データモデル要素に対して IEEE ドラフト規格[1]で定義された通り，固有のデータタイプを定義する • **値空間**: データタイプに保持される値の空間を表す．例えば，データタイプは文字列かもしれないが，文字列データ値は ASCII character set に限定されるかもしれない. • **フォーマット**: データタイプの値に対する制約のフォーマットを記述する．例えば，時間は特定のフォーマットを持つ(hh:mm:ss など) **LMS 動作要件:** • このセクションは LMS が遵守しなければならない要件を記述する **シーケンシングへの影響:**

	• このセクションは，シーケンシングルールおよび動作へのデータモデル要素の影響を記述する．このセクションが省かれた場合，データモデル要素はシーケンシングに何の影響も与えない **SCO 動作要件:** • このセクションは SCO が遵守しなければならない要件を記述する **API 実装要件:** • **GetValue():** このセクションは，特定のデータモデル要素に対する GetValue()要求を処理するとき LMS が遵守する固有の動作を記述する．また，このセクションは GetValue()要求で特定のデータモデル要素を利用したときに発生するエラー条件も記述する • **SetValue():** このセクションは，特定のデータモデル要素に対する SetValue()要求を処理するとき LMS が遵守する固有の動作を記述する．また，このセクションは，SetValue()要求で特定のデータモデル要素をと利用したときに発生するエラー条件も記述する **追加動作要件:** • このセクションは，データモデル要素に固有の追加動作要件を記述する **例:** • このセクションは，データモデル要素を用いる有効な API method コールの例を提供する

4.2.1. Version（データモデルバージョン）

cmi._version キーワードデータモデル要素は，LMS がサポートするデータモデルバージョンを SCO が決定するのに用いられる．SCORM ランタイム環境バージョン 1.3 に対する対応する cmi.version は 1.0 である．この値は IEEE 規格 [1]で定義される．この値は，LMS がサポートする SCORM ランタイム環境データモデルのバージョンを決定するために，SCO が要求することがある．例えば，SCO は，複数のバージョンの SCORM ランタイム環境データモデルをサポートするように作成することが可能である．

表 4.2.1a: Version データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi._version	データモデルのバージョンを表す **データモデル要素実装要件:** • **データタイプ**: 文字列 • **値空間**: ISO-10646-1 [5] • **フォーマット**: 値はピリオドで区切られた文字列から成り，整数のメジャーおよびマイナーリリース値を含む．マイナーリリース値の後に現れるキャラクターはマイナーリリース値からピリオド (“.”)によって区切られる [1]．文字列は IEEE ドラフト規格で定義された値を表す： “1.0” **LMS 動作要件:** • このデータモデル要素は必須で read-only として実装しなくてはならない • LMS は，SCO が要求したとき，文字列“1.0”を返す責任がある **SCO 動作要件:** • LMS はこのデータモデル要素を read-only として実装する必要がある．SCO は，データモデルバージョン (*cmi._version*) データモデル要素の値を読みだすことだけを許されている **API 実装要件:** • **GetValue():**LMS は “1.0” (上記の データモデル要素実装要件参照)を返しエラーコードを “0” – No Error に設定する • **SetValue():**SCO が *cmi._version* を設定する SetValue() 要求を呼び出すと，LMS はエラーコードを“404” – Data Model Element Is Read Only を設定し “false”を返却しなくてはならない **Example:** • GetValue(“cmi._version”)

4.2.2. Comments From Learner

コンテンツ設計者が，学習経験に関して学習者からコメントを集めたいと思うことがある．このデータモデルは，SCO 毎に学習者からのコメントを追跡する事を可能としている．どのようにコメントが集められるかもしくは提示されるかは SCORM の対象範囲外である．例えば，LMS は，LMS が提供するユーザーインターフェースコントロールを通して SCO へのコメントを集めるオプションを提供する事がある，もしくは，SCO が直接 SCO にコメントを集める機能をもつことがある．どのようにコメントが利用されるかも SCORM の対象範囲外である．例えば，一旦コメントが収集され，LMS が設計者にコメントをリストアウトする報告を作成する機能を提供することがある．これらは全体のコンテンツオーガニゼーションから集めることが可能である．これらのコメントは，設計者に現在の設計とコンテンツ構造を評価するために利用されることがある．

cmi.comments_from_learner データモデル要素は，コメントの実際のテキストだけでなく，ロケーションと時間を集める機能を提供する．

cmi.comments_from_learne は，学習者によって生成された書式自由のテキストを含む．テキストの構造は特定されていない．データモデル要素の値は，特定の学習者からの SCO に関する，SCO もしくは学習経験へのフィードバックを提供するように意図されている．このデータモデル要素を他の目的で利用する事は，相互運用可能性に悪影響を与えることがある．

LMS は学習者からの最低 250 コメントの SPM をサポートすべきである．学習者からの各コメントは，学習者によってなされたテキストのコメントを表すデータモデル要素を含む．このテキストのコメントは 4000 の SPM を持つ．LMS が SPM 以上の値をサポートすることは自由である．

表 4.2.2a: Comments From Learner データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.comments_from_learner._children	*cmi.comments_from_learner._children* データモデル要素はサポートされたデータモデル要素のリストを表す．このデータモデル要素は一般的に，どのデータモデル要素を LMS がサポートしているかを，SCO が決定するために用いられる．SCO が GetValue()および SetValue()要求に対して動的にパラメータを作るために，返却された文字列を用いることがある． **データモデル要素実装要件:** • **データタイプ**: 文字列 • **値空間**: ISO-10646-1 [5] • **フォーマット**: Comments From Learner 親データモデル要素において，LMS がサポートする全てのデータモデル要素のコンマで区切られたリスト．LMS は全てのデータモデル要素をサポートするように要求されているので，文字列は以下のデータのモデル要素を表さなくてはならない： o comment o location o timestamp **LMS 動作要件:**. • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS は，Comments From Learner の全ての子データモデル要素のコンマで区切られたリストを返却する

	責任がある（上記データモデル要素実装要件参照） **SCO 動作要件:** • LMS はこのデータモデル要素を read-only として実装する必要がある．SCO は，*cmi.comments_from_learner._children* データモデル要素の値を読みだすことだけが許されている **API 動作要件:** • **GetValue():** LMS は，LMS がサポートする Comments From Learner の子データモデル要素のコンマで区切られたリストを返却し（上記データモデル要素実装要件参照），エラーコードを “0” – No Error に設定する．データモデル要素の順序は重要ではない．返却された文字列はデータモデル要素実装要件で特定された要件を遵守しなくてはならない • **SetValue():** SCO が *cmi.comments_from_learner._children* を設定する SetValue()要求を呼びだした場合，LMS はエラーコードを“404” – Data Model Element Is Read Only に設定し“false”を返却しなくてはならない **Example:** • GetValue(“cmi.comments_from_learner._children”)
cmi.comments_from_learner._count	*cmi.comments_from_learner._count* キーワード は LMS が保管する学習者コメントの現在の数を SCO のために記述する．LMS が現在管理するエントリーの合計数を返却しなくてはならない． **データモデル要素実装要件:** • **データタイプ**: 整数 • **値空間**: 負でない整数 • **フォーマット**: LMS が現在管理する学習者コメントの数を表す文字列 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • SCO がどのコメントを設定するよりも前に，LMS が *cmi.comments_from_learner._count* 値を読みだす要求を受け取った場合，LMS は以下の API 実装要件に挙げられた要件を遵守しなくてはならない **SCO 動作要件:** • LMS はこのデータモデル要素を read-only として実装する必要がある．SCO は，*cmi.comments_from_learner._count* データモデル要素の値を読みだすことだけが許されている **API 実装要件:** • **GetValue():** LMS は，現在 LMS が保存する学習者コメントの数を返却し，エラーコードを “0” – No error に設定する o SCO によってコメントが設定されなければ，LMS は “0”を返却しエラーコードを “0” - No Error に設定する • **SetValue():** SCO が *cmi.comments_from_learner._count* を設定する SetValue()要求を呼び出した場合，LMS はエラーコードを“404” – Data Model Element Is Read

	Only に設定し“false”を返却しなくてはならない **例:** • GetValue(“cmi.comments_from_learner._count”)
cmi.comments_from_learner.n.comment	*cmi.comments_from_learner.n.comment* データモデル要素はテキストインプットを記述する [1]．文字列値はローカライズされた文字列を表す **データモデル要素要件:** • **データタイプ:** localized_string_type (SPM: 4000) • **値空間:** ローカル化情報を有する(ISO-10646-1 に定義された)文字列 • **フォーマット:** 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • 値は SCO が供給する．LMS はこのデータモデル要素に対して初期値を仮定してはならない．SCO がコメントを設定する前に GetValue()要求が行われたら，LMS は以下の API 動作要件に従って動作しなくてはならない **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は，*cmi.comments_from_learner.n.comment* データモデル要素の値を読みだしおよび書き込むことが許されている • SetValue()の際，デリミターはオプションであることに SCO は注意しなくてはならない．デリミターが文字列の 1 部として提供されない場合，LMS はデフォルト言語は “en” (英語)であると仮定する • GetValue()の際，デリミターが LMS が返却する文字列の一部であること（LMS の実装に依存する）に，SCO は注意しなくてはならない．LMS が返却した文字列を SCO がどのように扱うかは SCO の実装に依存する **API 実装要件:** • **GetValue():** LMS は，*cmi.comments_from_learner.n.comment* データモデル要素に保存された値を返却しエラーコードを “0” – No Error.に設定しなくてはならない．返却された文字列は，データモデル要素実装条件で指定された要件に遵守しなくてはならない o 集合に対するデータモデルバインディングは密な配列として表される． LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば(例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを “301” – General Get Failure に設定し，空の文字列 (“”)を返却しなくてはならない（セクション 3.1.7.6: *SCORM 拡張エラー条件* 参照） o SCO が

	cmi.comments_from_learner.n.comment を読み出そうとし，データのレコードは作成されているが，comment データモデル要素を SCO が設定していないとき，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し，空の文字列 (“”)を返却しなくてはならない • **SetValue():** LMS は *cmi.comments_from_learner.n.comment* を SetValue()要求で供給される値に設定し，エラーコードを “0” - No error に設定し “true”を返却しなくてはならない o SetValue()で供給された値がデータモデル要素実装要件の要件を満たさないなら，LMS はエラーコードを “406” – Data Model Element Type Mismatch に設定し “false”を返却しなくてはならない．LMS は，この要求に基づいてデータモデル要素の状態を変えてはならない o 集合に対するデータモデルバインディングは密な配列としてを表わされる．SCO が呼び出した SetValue()要求で提供されたインデックス (n)が保存された学習者コメントの現在の数（*cmi.comments_from_learner._count* を要求することで決定できる）より大きい場合，LMS はエラーコードを “351” – General Set Failure に設定し，“false”を返却しなくてはならない（セクション 3.1.7.6: *SCORM 拡張エラー条件* 参照） **追加動作要件:** • データモデル要素を実装する場合，SCO は *cmi.comments_from_learner.n.comment* が最初に順番に設定するようにしなくてはならない．SCO は前に設定されたコメント読み出して上書きし，コメント，ロケーション，タイムスタンプを更新することができる．SCO は LMS が実装する SPM は 4000 文字であることに注意しなくてはならない **Example:** • GetValue(“cmi.comments_from_learner.0.comment”) • SetValue(“cmi.comments_from_learner.0.comment”,”Some comments about the SCO”)
cmi.comments_from_learner.n.location	*cmi.comments_from_learner.n.location* データモデル要素は，コメントが適用される SCO 内での位置を示す．このデータモデル要素は各 SCO により実装に即して定義される．ロケーションに対して値が特定されない場合，コメントは SCO 中の特定の位置でなく SCO 全体に適用される [1] **データモデル要素実装要件:** • **データタイプ:** 文字列 (SPM: 250) • **値空間:** ISO-10646-1 • **フォーマット:** このデータモデル要素のフォーマットは SCO により定義されコントロールされる．より詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:**

	• このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • 値は SCO が管理し供給する．SCO がロケーションを設定する前に GetValue()要求が行われたら，LMS は以下の API 動作要件に従って動作しなくてはならない **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は，*cmi.comments_from_learner.n.location* データモデル要素の値を読みだしおよび書き込むことが許されている **API 実装要件:** • **GetValue():** LMS は *cmi.comments_from_learner.n.location* データモデル要素に保存された値を返却しエラーコードを“0” – No error に設定しねくてはならない．返却された文字列は，データモデル要素実装条件で指定された要件に遵守しなくてはならない o 集合に対するデータモデルバインディングは密な配列として表される． LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば(例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列 (“”)を返却しなくてはならない（セクション 3.1.7.6: *SCORM 拡張エラー条件* 参照） o SCO が *cmi.comments_from_learner.n.location* を読み出そうとし，データのレコードは作成されているが，comment データモデル要素を SCO が設定していないとき，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し，空の文字列 (“”)を返却しなくてはならない • **SetValue():** LMS は *cmi.comments_from_learner.n.location* を SetValue()要求で供給される値に設定し，エラーコードを “0” - No error に設定し “true”を返却しなくてはならない o 集合に対するデータモデルバインディングは密な配列として表わされる．SCO が呼び出した SetValue()要求で提供されたインデックス (n)が保存された学習者コメントの現在の数（*cmi.comments_from_learner._count* を要求することで決定できる）より大きい場合，LMS はエラーコードを “351” – General Set Failure に設定し，“false”を返却しなくてはならない（セクション 3.1.7.6: *SCORM 拡張エラー条件* 参照） **追加動作要件:** • SCO は前に設定されたコメント読み出して上書きし，コメント，ロケーション，タイムスタンプを更

	新することができる．SCO は LMS が実装する SPM は 250 文字であることに注意しなくてはならない **Example:** • GetValue(“cmi.comments_from_learner.0.location”) • SetValue(“cmi.comments_from_learner.0.location”,”PAGE1SECTION#3”)
cmi.comments_from_learner.n.timestamp	*cmi.comments_from_learner.n.timestamp* データモデル要素は，コメントが作成もしくは最も最近に変更された時点を示す．実装は，少なくとも 1970 年 1 月 1 日から 2038 年 1 月 1 日までの期間をサポートする [1] **データモデル要素実装要件:** • **データタイプ**: time (second,10,0) • **値空間**: データタイプは精度 1 秒およびオプションの 100 分の 1 秒で表現された時間の値を表す • **フォーマット**: より詳細はセクション 4.1.1.7: *データタイプ*を参照 **LMS 実装要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • 値は SCO が管理し供給する．SCO がタイムスタンプを設定する前に GetValue()要求が行われたら，LMS は以下の API 動作要件に従って動作しなくてはならない **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は，*cmi.comments_from_learner.n.timestamp* データモデル要素の値を読みだしおよび書き込むことが許されている **API 実装要件:** • **GetValue():** LMS は *cmi.comments_from_learner.n.timestamp* データモデル要素に保存された値を返却しエラーコードを“0” – No error に設定しなくてはならない．返却された文字列は，データモデル要素実装条件で指定された要件に遵守しなくてはならない o 集合に対するデータモデルバインディングは密な配列として表される． LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば(例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを “301” – General Get Failure に設定し，空の文字列 (“”)を返却しなくてはならない（セクション 3.1.7.6: *SCORM 拡張エラー条件* 参照） o SCO が *cmi.comments_from_learner.n.timestamp* を読み出そうとし，データのレコードは作成されているが，*cmi.comments_from_learner.n.timestamp* データモデル要素を SCO が設定していないとき，LMS はエラーコードを“403” Data

	Model Element Value Not Initialized に設定し，空の文字列 (“”)を返却しなくてはならない • **SetValue():** LMS は *cmi.comments_from_learner.n.timestamp* を SetValue() 要求で供給される値に設定し，エラーコードを “0” - No error に設定し “true”を返却しなくてはならない o SetValue()で供給された値がデータモデル要素実装要件の要件を満たさないなら，LMS はエラーコードを “406” – Data Model Element Type Mismatch に設定し “false”を返却しなくてはならない．LMS は，この要求に基づいてデータモデル要素の状態を変えてはならない o 集合に対するデータモデルバインディングは密な配列として表わされる．SCO が呼び出した SetValue()要求で提供されたインデックス (n)が保存された学習者コメントの現在の数（*cmi.comments_from_learner._count* を要求することで決定できる）より大きい場合，LMS はエラーコードを “351” – General Set Failure に設定し，“false”を返却しなくてはならない（セクション 3.1.7.6: *SCORM 拡張エラー条件参照*） **例:** • GetValue(“cmi.comments_from_learner.0.timestamp”) • SetValue(“cmi.comments_from_learner.0.timestamp”,“2003-07-25T03:00:00”)

4.2.3. Comments From LMS

cmi.comments_from_lms データモデル要素は，SCO のすべての学習者が見ることを意図したコメントや注釈からなる．これらのコメントは，特定のコミュニティや講師ノートなどから，全ての学習者の関心となる情報を追加するメカニズムとなるように意図されている．SCORM は，これらのコメントがどのように初期化されるかに関するメカニズムは定義しない．LMS は，このデータの作成および初期化をサポートするメカニズムを自由に提供できる．このサポートは SCORM 準拠のためには要求されていない．

この情報がどのように提示されて利用されるかは SCORM の対象範囲外である．そのような使用法の一つは，コメントを読み出し，SCO の起動と同時に（もしくは学習セッションのある 1 時点で）学習者にそのコメントを提示する機能である．

cmi.comments_from_lms の値は SCO もしくは SCO を用いた学習経験に関する情報を提供するように意図されている．このデータの構造は SCORM によって特定されていない．

LMS は，LMS からの最低 100 コメントの SPM をサポートする．LMS がそれ以上の SPM をサポートするのは自由である．

表 4.2.3a: Comments From LMS データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.comments_from_lms._children	*cmi.comments_from_lms._children* データモデル要素は，サポートされたデータモデル要素のリストを表す．このデータモデル要素は一般的に，どのデータモデル要素を LMS がサポートしているかを，SCO が決定するために用いられる． SCO が GetValue()および SetValue()要求に対して動的にパラメータを作るために，返却された文字列を用いることがある． **データモデル要素実装要件:** • **データタイプ**: 文字列 • **値空間**: ISO-10646-1 [5] • **フォーマット:** Comments From LMS 親データモデル要素において，LMS がサポートする全てのデータモデル要素のコンマで区切られたリスト．LMS は全てのデータモデル要素をサポートするように要求されているので，文字列は以下のデータのモデル要素を表さなくてはならない： o comment o location o timestamp **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS は，Comments From Learner の全ての子データモデル要素のコンマで区切られたリストを返却する責任がある（上記データモデル要素実装要件参照） **SCO 動作要件:** • LMS はこのデータモデル要素を read-only として実装する必要がある．SCO は，

	cmi.comments_from_lms._children データモデル要素の値を読み出すことだけが許されている **API 動作要件:** • **GetValue():** LMS は，LMS がサポートするデータモデル要素のコンマで区切られたリストを返却し（上記データモデル要素実装要件参照），エラーコードを "0" – No Error に設定する．データモデル要素の順序は重要ではない．返却された文字列はデータモデル要素実装要件で特定された要件を遵守しなくてはならない • **SetValue():** SCO が *cmi.comments_from_lms._children* を設定する SetValue()要求を呼び出だした場合，LMS はエラーコードを"404" – Data Model Element Is Read Only に設定し"false"を返却しなくてはならない **Example:** • GetValue("cmi.comments_from_lms._children")
cmi.comments_from_lms._count	*cmi.comments_from_lms._count* キーワードは，LMS が保管する LMS からのコメントの現在の数を記述する．LMS が現在管理するエントリーの合計数を返却しなくてはならない． **データモデル要素実装要件:** • **データタイプ**: 整数 • **値空間**: 負でない整数 • **フォーマット**: LMS が現在保持するコメントの数を表す文字列 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • SCO がどのコメントを設定するよりも前に，LMS が *cmi.comments_from_lms._count* 値を読みだす要求を受け取った場合，LMS は以下の API 実装要件に挙げられた要件を遵守しなくてはならない **SCO 動作要件:** • LMS はこのデータモデル要素を read-only として実装する必要がある．SCO は，*cmi.comments_from_lms._count* データモデル要素の値を読みだすことだけが許されている **API 実装要件:** • **GetValue():** LMS は，現在 LMS が保存する学習者コメントの数を返却し，エラーコードを "0" – No error に設定する o この要素に対してコメントが定義されていなければ，LMS は "0"を返却しエラーコードを "0" - No Error に設定する • **SetValue():** SCO が *cmi.comments_from_lms._count* を設定する SetValue()要求を呼び出せば，LMS はエラーコードを"404" – Data Model Element Is Read Only に設定し"false"を返却しなくてはならない **例:**

	• GetValue("cmi.comments_from_lms._count")
cmi.comments_from_lms.n.comment	*cmi.comments_from_lms.n.comment* データモデル要素は SCO に対するコメントと注釈を記述する [1]．文字列値はローカライズされた文字列を表す **データモデル要素要件:** • **データタイプ**: localized_string_type (SPM: 4000) • **値空間**: ローカル化情報を有する(ISO-10646-1 に定義された)文字列 • **フォーマット**: 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • どのようにこのデータモデル要素が初期化されるかは SCORM の対象範囲外である．LMS は，作成者/インストラクターがコメントを提供することを可能にし，これらのコメントはこの値を初期化するために用いられることがある．コメントが設定される前もしくは LMS に初期化される前に GetValue()要求が行われたら，LMS は以下の API 動作要件に従って動作しなくてはならない **SCO 動作要件:** • LMS はこのデータモデル要素を read-only として実装する必要がある．SCO は，*cmi.comments_from_lms.n.comment* データモデル要素の値を読みだすことだけが許されている • GetValue()の際，デリミターが LMS が返却する文字列の一部であること（LMS の実装に依存する）に，SCO は注意しなくてはならない．LMS が返却した文字列を SCO がどのように扱うかは SCO の実装に依存するデリミターが文字列の一部として提供されない場合，SCO はデフォルト言語は "en" (英語)であると仮定する **API 実装要件:** • **GetValue()**: LMS は，*cmi.comments_from_lms.n.comment* データモデル要素に保存された値を返却しエラーコードを "0" – No Error.に設定しなくてはならない．返却された文字列は，データモデル要素実装条件で指定された要件に遵守しなくてはならない o 集合に対するデータモデルバインディングは密な配列として表される． LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば(例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを "301" – General Get Failure に設定し，空の文字列 ("")を返却しなくてはならない（セクション 3.1.7.6: *SCORM 拡張エラー条件参照*) o SCO が *cmi.comments_from_lms.n.comment* を読み

	出そうとし，データのレコードは作成されているが，comment データモデル要素を LMS が設定していないとき，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し，空の文字列 (“”)を返却しなくてはならない • **SetValue():** *cmi.comments_from_lms.n.comment* を設定するために SCO が SetValue()要求を呼び出したら，LMS はエラーコードを “404” - Data Model Element Is Read Only に設定し “false”を返却しなくてはならない．LMS はこの要求に基づいてデータモデル要素の状態を変えてはならない **例:** • GetValue(“cmi.comments_from_lms.0.comment”)
cmi.comments_from_lms.n.location	*cmi.comments_from_lms.n.location* データモデル要素は，コメントが適用される SCO 内での位置を示す．このデータモデル要素は各 SCO により実装に即して定義される．ロケーションに対して値が特定されないなら，コメントは SCO 中の特定の位置でなく SCO 全体に適用する [1] **データモデル要素実装要件:** • **データタイプ**: 文字列 (SPM: 250) • **値空間**: ISO-10646-1 • **フォーマット**: このデータモデル要素のフォーマットは SCO により定義されコントロールされる．より詳細はセクション 4.1.1.7: *データタイプ* 参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • どのようにこのデータモデル要素が初期化されるかは SCORM の対象範囲外である．LMS は，作成者/インストラクターがコメントを提供することを可能にし，これらのコメントはこの値を初期化するために用いられることがある．コメントが設定される前もしくは LMS に初期化される前に GetValue()要求が行われたら，LMS は以下の API 動作要件に従って動作しなくてはならない **SCO 動作要件:** • LMS はこのデータモデル要素を read-only として実装する必要がある．SCO は，*cmi.comments_from_lms.n.location* データモデル要素の値を読みだすことだけが許されている **API 実装要件:** • **GetValue():** LMS は *cmi.comments_from_lms.n.location* データモデル要素に保存された値を返却しエラーコードを“0” – No error に設定する．返却された文字列は，データモデル要素実装条件で特定された要件に遵守しなくてはならない o 集合に対するデータモデルバインディ

	ングは密な配列として表される．LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば(例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを “301” – General Get Failure に設定し，空の文字列 (“”)を返却しなくてはならない（セクション 3.1.7.6: *SCORM 拡張エラー条件* 参照） o SCO が *cmi.comments_from_lms.n.location* を読み出そうとし，データのレコードは作成されているが，location データモデル要素を LMS が設定していないとき，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し，空の文字列 (“”)を返却しなくてはならない • **SetValue():** *cmi.comments_from_lms.n.location* を設定するために SCO が SetValue()要求を呼び出したら，LMS はエラーコードを “404” - Data Model Element Is Read Only に設定し “false”を返却しなくてはならない．LMS はこの要求に基づいてデータモデル要素の状態を変えることてはならない **例:** • GetValue(“cmi.comments_from_lms.0.location”)
cmi.comments_from_lms.n.timestamp	*cmi.comments_from_lms.n.timestamp* データモデル要素は，コメントが作成もしくは最も最近に変更された時点を示す．実装は，最低 1970 年 1 月 1 日から 2038 年 1 月 1 日までの期間をサポートする [1] **データモデル要素実装要件:** • **データタイプ:** time (second,10,0) • **値空間:** データタイプは精度 1 秒およびオプションの 100 分の 1 秒で表現された時間の値を表す．時間値における秒の数は 1970 年 1 月 1 日の 00:00 からの秒の数である[1] • **フォーマット:** より詳細はセクション 4.1.1.7: *データタイプ* を参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • どのようにこのデータモデル要素が初期化されるかは SCORM の対象範囲外である．LMS は，作成者/インストラクターがコメントを提供することを可能にし，これらのコメントはこの値を初期化するために用いられることがある．コメントが設定される前もしくは LMS に初期化される前に GetValue()要求が行われたら，LMS は以下の API 動作要件に従って動作しなくてはならない **SCO 動作要件:** • LMS はこのデータモデル要素を read-only として

	実装する必要がある．SCO は，*cmi.comments_from_lms.n.timestamp* データモデル要素の値を読みだすことだけが許されている **API 実装要件:** • **GetValue():** LMS は *cmi.comments_from_lms.n.timestamp* データモデル要素に対して保存された値を返却しエラーコードを“0” – No error に設定しなくてはならない．返却された文字列は，データモデル要素実装条件で指定された要件に遵守しなくてはならない o 集合に対するデータモデルバインディングは密な配列として表される． LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば(例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを “301” – General Get Failure に設定し，空の文字列 (“”)を返却しなくてはならない（セクション 3.1.7.6: *SCORM 拡張エラー条件* 参照） o SCO が *cmi.comments_from_lms.n.timestamp* を読み出そうとし，データのレコードは作成されているが，timestamp データモデル要素を LMS が設定していないとき，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し，空の文字列 (“”)を返却しなくてはならない • **SetValue():** *cmi.comments_from_lms.n.timestamp* を設定するために SCO が SetValue()要求を呼び出したら，LMS はエラーコードを “404” - Data Model Element Is Read Only に設定し “false”を返却しなくてはならない．LMS はこの要求に基づいてデータモデル要素の状態を変えてはならない **例:** • GetValue(“cmi.comments_from_lms.0.timestamp”)

4.2.4. Completion Status

cmi.completion_status データモデル要素は，学習者が SCO を完了したかどうかを示す[1]．SCO がどのように完了ステータスを決定するかは SCORM の対象範囲外である．この値は，SCO 開発者が定義した SCO の全体完了ステータスを示す．

表 4.2.4a: Completion Status データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.completion_status	**データモデル要素実装要件:** • **データタイプ**: 状態 (completed, incomplete, not_attempted, unknown) • **値空間**: IEEE ドラフトは 4 つの状態値を定義する．SCORM はこれらの状態値を以下の制限された語彙トークンにバインディングする： o “completed”: SCO が完了したと考えられる程度充分に学習者が SCO を学習した [1]．どのように完了を決定するかは SCO が制御および管理する．完了状態は，4.2.4.1:*完了状態の決定*で定義される要件に基づいて LMS によって上書きされる o “incomplete”: SCO が完了したと考えられるほど充分に学習者が SCO を学習していない [1]．どのように完了を決定するかは SCO が制御および管理する． o “not attempted”: 学習者が SCO を何らかの形で使用したと考えられない [1] o “unknown”: 根拠がない [1]．これは，完了状態に関する適用可能な情報がないことを示す • **フォーマット**: データモデル値のフォーマットは，4 つの限定された語彙トークンの一つである (“completed”, “incomplete”, “not attempted”, “unknown”) **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • *cmi.completion_threshold* が定義されないなら, *cmi.completion_status* の定義は SCO によってコントロールされ管理される. LMS は *cmi.completion_status* へのどんな値も示すことができない. SCO が *cmi.completion_status* を設定しないなら, LMS は *cmi.completion_status* の値としてデフォルトの"unknown"を使う. • *cmi.completion_threshold* が定義されるなら, *cmi.completion_threshold, cmi.progress_measure,* および *cmi.completion_status* のために LMS が用いる値の整合性を保つことは LMS の責任となる. LMS は(GetValue() 呼び出しで要求されたら)セクション 4.2.4.1 *Completion Status Evaluation* にて定義された要求に従い *cmi.completion_status* を報告しなくてはならない. **シーケンシングへの影響:** • (セクション 4.2.4.1 で記述されたプロセスを通して)SCO もしくは LMS が SCO の *cmi.completion_status* を“unknown”へ設定すると，SCO に関連付けられた学習アクティビティの Attempt Progress Status は false となる • (セクション 4.2.4.1 で記述されたプロセスを通して)SCO もしくは LMS が SCO の *cmi.completion_status* を“completed”へ設定すると，SCO に関連付けられた学習アクティビティの Attempt Progress Status

	は true となり，SCO に関連付けられた学習アクティビティの Attempt Completion Status は true となる • (セクション 4.2.4.1 で記述されたプロセスを通して)SCO もしくは LMS が SCO の *cmi.completion_status* を“incomplete”へ設定すると，SCO に関連付けられた学習アクティビティの Attempt Progress Status は true となり，SCO に関連付けられた学習アクティビティの Attempt Completion Status は false となる **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は，*cmi.completion_status* データモデル要素の値を読みだしおよび書き込むことが許されている • *cmi.completion_status* を設定すると SCO に関連付けられた学習アクティビティに影響があり，従ってシーケンシングにも影響する可能性があることに留意するべきである • SCO に関連付けられた学習アクティビティに適用されるシーケンシング情報で完了状態に依存するものがあるならば，SCO の学習者セッションが終わる前に，SCO は完了情報が正確に LMS に送られたことを保証しなければならない．そうでないと，シーケンシング情報を処理するとき，LMS は SCO と関連付けられた学習アクティビティの完了状態として “unknown”値を利用する **API 実装要件:** • **GetValue():** LMS は学習者に関して LMS が現在格納している *cmi.completion_status* を返すしエラーコードを“0” – No error に設定する．返却された文字列はデータモデル要素実装要件で指定された要件を遵守する o *cmi.completion_threshold* が定義されるなら, LMS は (GetValue()呼び出しで要求されたら)セクション 4.2.4.1 *Completion Status Evaluation* にて定義された要求に従い *cmi.completion_status* を報告すべきである. o *cmi.completion_threshold* が定義されないなら, SCO が値を報告するまで, *cmi.completion_status* のデフォルト値は "unknown"である. • **SetValue():** SCO が *cmi.completion_status* を設定する要求を呼び出し，値が上記の限定された語彙トークンのいずれかでない場合，LMS は API エラーコードを“406” – Data Model Element Type Mismatch に設定し “false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えてはならない **例:** • GetValue(“cmi.completion_status”) • SetValue(“cmi.completion_status”,”incomplete”)

4.2.4.1 Completion Status の決定

一般的に，SCO の完了状態は，SCO によって決定される．SCO の完了状態は，コンテンツ開発者によって様々なやり方で設計される．これらのやり方は，以下に限定されることはないが，以下を含むようなものである：

- 学習者が訪れたページの数
- 学習者が選択するユーザーインターフェースコントロールの結果 (学習者がボタンを押すなど)
- 学習者がビデオを見ることもしくはドキュメントを読むことの完了
- SCO の様々な学習目標の完了(すなわち cmi.objectives.*n*.completion_status)

SCO によってどのように決定が成されるかに関わらず，このプロセスは SCO が cmi.completion_status を設定することを含む．SCORM では，SCO が SCORM ランタイム環境データモデル要素を追跡すること

(GetValue()もしくは SetValue()関数呼び出し)を要求しない．これを念頭に，LMS は完了状態の決定に要求される追加の動作を行う．ある特定の状況の下で，LMS は異なった動作を要求される．

cmi.completion_status データモデル要素は，二つの他の SCORM ランタイム環境データモデル要素に影響される (完了閾値 – cmi.completion_threshold および進捗度 – cmi.progress_measure)．以下の表は，これらの値の状態および定義された LMS 動作を規定する．完了閾値は，imsmanifest.xml ファイルで定義されることがある(セクション 4.2.5:*完了閾値*参照)．進捗度および完了状態はともに SCO が決定し設定する．表 4.2.4.1a は設定された，または設定されないこれらの値の組み合わせと関連する LMS の動作を定義する．LMS の動作は，cmi.completion_status の値を取得するための GetValue()要求が SCO によって発せられた時に適用される．LMS に格納された cmi.completion_status の実際の値が上書きされたり決定の過程で持続するかどうか，は SCORM 規格の範囲外であり実装依存である．

表 4.2.4.1a: Completion Status 決定

完了閾値	進捗度	完了状態	LMS 動作
未定義	SCO による設定値なし	SCO による設定値なし	cmi.completion_status を“unknown”に設定しなくてはならない
未定義	SCO による設定値なし	定義された語彙の一つ	何もしない．SCO が設定した値が cmi.completion_status に設定される
未定義	0.5	定義された語彙の一つ	何もしない．SCO が設定した値が cmi.completion_status に設定される
0.8	0.5	定義された語彙の一つ	cmi.completion_status を上書きし“incomplete”に設定しなくてはならない 根拠： 0.5 < 0.8. cmi.completion_threshold および cmi.progress_measure で定義される要件参照
0.8	0.9	定義された語彙の一つ	cmi.completion_status を上書きし“complete”に設定しなくてはならない 根拠：0.9 > 0.8. cmi.completion_threshold および cmi.progress_measure で定義される要件参照
0.8	SCO による設定値なし	SCO による設定値なし	cmi.completion_status を“unknown”に設定しなくてはならない
0.8	0.5	SCO による設定値なし	cmi.completion_status を“incomplete”に設定しなくてはならない 根拠： 0.5 < 0.8. cmi.completion_threshold および cmi.progress_measure で定義される要件参照
0.8	0.9	SCO による設定値なし	cmi.completion_status を“complete”に設定しなくてはならない 根拠：0.9 > 0.8. cmi.completion_threshold および cmi.progress_measure で定義される要件参照
未定義	0.5	SCO による設定値なし	cmi.completion_status を“unknown”に設定しなくてはならない
0.8	SCO による設	定義された語彙	何もしない．SCO が設定した値が

	定値なし	の一つ	cmi.completion_status に設定される

ADL Note: Completion Threshold は, SCO リソースを参照する<adlcp:item>要素と関連する ADL 名前空間の要素<adlcp:completionThreshold>によって定義される. <adlcp:completionThreshold>が定義されるなら, LMS は cmi.completion_threshold データモデル要素が使われる値に初期化する責任を負う. <adlcp:completionThreshold>が定義されないなら cmi.completion_threshold は定義されず, LMS によってどんな値に初期化されることもない.

4.2.5. Completion Threshold

cmi.completion_threshold データモデル要素に保存された値は，SCO が完了したとみなされるかどうかを決定するのに利用される．その SCO の完了について，cmi.completion_threshold と学習者による `cmi.progress_measure` **を比較することで決定される．**

表 4.2.5a: Completion Status データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.completion_threshold	**データモデル要素実装要件:** • **データタイプ:** 実数(10,7) range (0..1) • **値空間:** 0.0 <= *cmi.completion_threshold* <= 1.0 • **フォーマット:** より詳細はセクション 4.1.1.7: *データタイプ SCORM 拡張エラー条件* **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS は，ADL Content Packaging namespace element <adlcp:completionThreshold>を利用して，このデータモデル要素を初期化する責任がある．この要素は，必要があるときは，コンテンツパッケージマニフェストで見つかる SCO リソースを参照する<imscp:item>要素にだけおかれる **SCO 動作要件:** • LMS はこのデータモデル要素を read-only として実装する必要がある．SCO は *cmi.completion_threshold* データモデル要素の値を読み出すことだけが許されている **API 動作用件:** • **GetValue():** LMS は，*cmi.completion_threshold* データモデル要素に保存された値を返却し，エラーコードを“`0`” – No error に設定する．返却された文字列はデータモデル要素実装要件で指定された要件を遵守しなくてはならない o SCO attempts to retrieve the *cmi.completion_threshold* を読み出そうととし，コンテンツパッケージマニフェストで completion threshold が定義されていないなら，LMS はエラーコードを “`403`” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返却しなくてはならない • **SetValue():** *cmi.completion_threshold* を設定するために SCO が SetValue()要求を呼び出せば，LMS はエラーコードを “`404`” – Data Model Element Is Read Only に設定し “`false`”を返却しなくてはならない．LMS はこの要求に基づいてデータモデル要素の状態を変えてはならない **例:** • GetValue(“cmi.completion_threshold”)

4.2.6. Credit

cmi.credit データモデル要素は，学習者が SCO でのパフォーマンスに対して評価を与えられたかどうかを示す [1]．SCO がどのように credit や no credit を指定するよう規定するかについては，SCORM の対象範囲外である．このデータモデル要素のデフォルト値は，credit に対してとられるものである。

一般的に，人はどの SCO がクレジットの対象となるかを選択しない．これは，通常コンテンツオーガニゼーション（コース等）レベルで処理される．学習者が credit，no credit をコンテンツオーガニゼーションで登録することを許すメカニズムを LMS が提供するならば，学習者の選択した値はコンテンツオーガニゼーションにある全ての SCO で用いられる．

表 4.2.6a: Credit データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.credit	**<u>データモデル要素実装要件:</u>** • ***データタイプ*: 状態**(credit, no_credit) • **値空間:** IEEE ドラフトは二つの状態値を定義する．SCORM はこれらの状態値を以下の制限された語彙トークンに定める ： o “credit”: 学習者は SCO の評価をとる [1]． このデータモデル要素のデフォルト値は “credit”である．この値を決定もしくは与えるメカニズムが供給されないと， デフォルト値の “credit”が利用される．SCO の評価をとる事は，*cmi.success_status* データモデル要素の決定に影響する (セクション 4.2.16.1: *モードおよびクレジット利用要件*参照) o “no-credit”: 学習者は SCO の評価をとらないとする [1]．学習者が SCO の評価をとらないとすれば，success status もしくは score の決定に影響する値は，LMS に解釈されず，SCO に対する現在の success status もしくは score に反映されない • **フォーマット:** このデータモデル値のフォーマットは，上記の二つの制限されたトークンのうちどちらかである(“credit”, “no-credit”) **<u>LMS 動作要件:</u>** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • SCORM は，LMS が SCO に対するクレジットの状態を割り振るメカニズムをサポートすることを要求しない．LMS がこのようなメカニズムをサポートするか否かは SCORM の対象範囲外である．LMS がメカニズムをサポートしない場合，クレジットのデフォルト値 (“credit”)が全てのケースで返却されなくてはなれない **<u>SCO 動作要件:</u>** • LMS はこのデータモデル要素を read-only として実装する必要がある．SCO は *cmi.credit* データモデル要素の値を読み出すことだけが許されている • このデータと SCO が何をするかは完全に SCO に任されている．SCO は，他のデータモデル要素値を LMS にレポートすることの重要性を決定するために *cmi.credit* を利用する事が可能である **<u>API 実装要件:</u>** • **GetValue():** LMS は学習者に対して現在 LMS に保存されている *cmi.credit* を返却しエラーコードを“0” – No error に設定する．返却さ

	れた文字列はデータモデル要素実装要件に指定された要件を遵守しなくてはならない o SCO が *cmi.credit* を要求し，LMS が “credit” vs. “no-credit”を選択するメカニズムをサポートしない場合，LMS はデフォルト値(“credit”)を返却しなくてはならない o LMS がそのようなメカニズムもしくはプロセスをサポートする場合，LMS は *cmi.credit* に対して現在保存された値を返却しなくてはならない • **SetValue():** *cmi.credit* を設定するために SCO が要求を呼び出せば，LMS は API エラーコードを“404” – Data Model Element Is Read Only に設定し “false”を返却しなくてはならない．LMS はこの要求に基づいてデータモデル要素の状態を変えてはならない. **Example:** • GetValue(“cmi.credit”)

4.2.7. Entry

cmi.entry データモデル要素は，学習者が以前 SCO にアクセスしたかどうかを主張する情報を含む [1].

時間モデル(セクション 2.1.1: Run-Time Environment Temporal Model 参照)で定義されたように，エントリー値：“ab-initio”は，SCO はデフォルト（空）のランタイムデータを持つことを示す– 前回の学習者のアテンプトから利用可能なランタイムデータはない．エントリー値：“resume”は，SCO が現在の学習者のアテンプトに対して，前回の学習セッションで設定されたランタイムデータにアクセスしていることを示す．

表 4.2.7a: Entry データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.entry	**<u>データモデル要素実装要件:</u>** • ***データタイプ*: 状態** (ab_initio, resume, _nil_) [1] • **値空間**: IEEE ドラフトは二つの状態値を定義する．SCORM はこれらの状態値を以下の制限された語彙トークンに定める ： o “ab-initio”: 学習者が現在の学習者のアテンプト中の SCO にアクセスしていないことを示す [1]．この値は，これは SCO への現在の学習者のアテンプトと関連する最初の学習セッションであることを示す o “resume”: (1) 学習者が現在の学習者のアテンプト中，学習者が以前 SCO にアクセスしたことを示す，および(2)エグジット時に *cmi.exit* データモデル要素が “suspend” もしくは “logout”の値であったことを示す [1] o “” (空の文字列): 他の全ての条件を示す．前回アクセスの認識がない，もしくは指定のエントリー条件が示されていない [1]．もう一つの例は，SCO が既に完了もしくはマスターされており，後で復習目的で起動される **<u>LMS 動作要件:</u>** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS は以下のルールに基づいて *cmi.entry* データモデル要素を初期化する責任がある： o これが学習者のアテンプトの最初の学習セッションなら，SCO の最初の起動と同時に LMS は *cmi.entry* を “ab-initio”へ設定する． o SCO への学習者のアテンプトが中断され、学習セッションが再開されたら，LMS はこの値を“resume”に初期化する o SCO の前回の学習セッションが LMS の発する Suspend All ナビゲーション要求によって終了したならば, LMS はこの値を "resume"に初期化する. o 他の全ての条件に対して，LMS は *cmi.entry* を空の文字列に設定する • Terminate(“”)要求を受け取るもしくはユーザーが離れると同時に，LMS は *cmi.entry* を空の文字列（“” ）か“resume”に設定する．この値は現在の学習者のアテンプトで次の学習セッション中に利用可能となる．これは，*cmi.exit* の値を評価することにより LMS に決定される．アクティビティのアテンプトが中断されると，LMS は，アクティビティの次のアテンプトと同時に *cmi.entry* を “resume”へ設定する．SCO が *cmi.exit* を“suspend”以外の値に設定すると，もしくは，値を全

	く設定しないと，LMS は *cmi.entry* を空の文字列へ設定する **SCO 動作要件:** • LMS はこのデータモデル要素を read only として実装する必要がある．SCO は *cmi.entry* データモデル要素の値を読み出すことだけが許されている • このデータと SCO が何をするかは完全に SCO に任されている **API 実装要件:** • **GetValue():** LMS は学習者に対して現在 LMS に保存されている *cmi.entry* 値を返却しエラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件に指定された要件を遵守しなくてはならない • **SetValue():** *cmi.entry* を設定するために SCO が要求を呼び出せば，LMS は API エラーコードを“404” – Data Model Element Is Read-Only に設定し “false”を返却しなくてはならない．LMS はこの要求に基づいてデータモデル要素の状態を変えてはならない． **例:** • GetValue(“cmi.entry”)

4.2.8. Exit

cmi.exit データモデル要素は，どのようにもしくはなぜ学習者が SCO を離れたかを示す．この値は，SCO が最後にエグジットされた理由を示す．

cmi.exit データモデル要素は，SCO のランタイム実行の一時的な様相と関係している．

- cmi.exit が"suspend"に設定されたら，SCO の現在の学習者アテンプトは終了する．SCO が以降の学習セッションで再起動されるなら，現在の学習セッションについての SCO のランタイム環境のデータモデル要素の値は SCO によって利用可能である．
- cmi.exit が"normal","logout","time-out"または""(空文字列)に設定されるなら，SCO の学習アテンプトは終了する．現在の学習セッションについての SCO のランタイム環境のデータモデル要素の値は SCO が再起動されても利用不可となる．

ADL Note: LMS が Suspend All ナビゲーション要求を発動するならば，cmi.exit の値は無視される．これらの場合，SCO の現在の学習アテンプトは終了しない．SCO のランタイム環境のデータモデル要素の値は保持され，SCO が再起動されれば SCO から利用可能となる．

表 4.2.8a: Exit データモデル要素に対する Dot-notation バインディング

Dot-Notation Binding	Details
cmi.exit	**データモデル要素実装要件:** • **データタイプ:** 状態 (timeout, suspend, logout, normal, _nil_) • **値空間:** IEEE ドラフトは 5 個の値を定義する．SCORM はこれらの状態値を以下の制限された語彙トークンに定める： o “time-out”: max_time_allowed に指定された制限時間を越えたので SCO が終了した [1] o “suspend”: エグジットの時点では，学習者が後に SCO へ戻る意図を持って SCO をエグジットした [1] o “logout”: SCO が SCO が 1 部である学習アクティビティを終了する要望を示した o “normal”: SCO が正常にエグジットした [1] o “” empty characterstring: エグジット条件が指定されない [1]．デフォルト値 **LMS 動作要件:** • このデータモデル要素は必須であり LMS は write-only として実装しなくてはならない • 値は完全に SCO にコントロールされる．SCO はこの値を設定する責任がある．SCO が cmi.exit データモデル要素を設定しないとデフォルト値(空の文字列– “”)が用いられる．LMS が *cmi.exit* 値を得る要求を受け取れば，LMS は下記の API 実装要件を遵守しなくてならない **シーケンシングインパクト:** • SCO が *cmi.exit* を“time-out”に設定する場合は，LMS は SCO を取り去り，保留中のナビゲーション要求に関わらず，“Exit All”ナビゲーション要求を処理する • SCO が *cmi.exit* を“suspend”に設定すると，LMS はその SCO に結ばれる学習アクティビティの Activity is Suspended 値を true に設定する • SCO が *cmi.exit* を“logout”に設定すると，LMS は SCO を退け，未決定のナビゲーション要求に代わって，“Suspend All”ナビゲーション要求を処理する **SCO 動作要件:**

- LMS はこのデータモデル要素を write-only として実装する必要がある．SCO は *cmi.exit* データモデル要素の値を保存することだけが許可されている

API 実装要件:

- **GetValue():** SCO が *cmi.exit* を得る要求を呼び出せば，LMS はエラーコードを“405” – Data Model Element Is Write Only に設定し空の文字列(“”)を返却しなくてはならない
- **SetValue():** この要求は *cmi.exit* を供給された値へ設定する．この値は，このデータモデル要素に対応する制限された語彙の一つとマッチしなければならない．供給された値が上記のトークンの一つとマッチすると，LMS は値を設定し“true”を返却しエラーコードを“0” - No error に設定する
    - 供給された値が上記のトークンの一つとマッチしないと LMS はエラーコードを“406” - Data Model Element Type Mismatch に設定し“false”を返却しなくてはならない．LMS はこの要求に基づいてデータモデル要素の状態を変えてはならない

追加動作要件:

SCO への各学習者のアテンプトに対し，その学習者のアテンプトの最初の学習セッションで，*cmi.entry* 値の“ab-initio”を利用可能にするのは LMS の責任である．この値は，SCO へのこのアテンプト中，これが学習者が SCO を経験する最初であることを示す－またデフォルトデータモデルが SCO にアクセスされたことも示す．*cmi.entry* 値は read-only として実装される．このデータ要素は read-only として実装されるので，LMS はこの値を管理する責任がある（SCO は直接この値に影響しない－*cmi.entry* に SetValue()を呼び出せない）．LMS はこの値を決定するのに他のランタイムに連動する責任がある．詳細は以下に説明される：

cmi.entry 値は直接 *cmi.exit* データモデル要素に影響される．SCO は *cmi.exit* 値を設定する責任がある．*cmi.exit* の値によって LMS は以下のように行動する：

1. SCO が *cmi.exit* を“time-out”に設定すると，学習者のアテンプトは終了することと仮定される．学習者のアテンプトが終了したので，次回 SCO が起動されると，LMS はクリーンなデータモデル値を供給する．更に，LMS は学習者に対してコンテンツオーガニゼーションへのアテンプトを終了する．この動作は，Terminate(“”)要求が SCO から受け取られたときに，もしくは学習者が退出したときに提示される．
2. SCO が *cmi.exit* を“suspend”に設定すると，LMS は *cmi.entry* を“resume”へ設定する．*cmi.exit* を“suspend”に設定することにより，SCO は，学習者が後で SCO に戻る意図を持って SCO をエグジットしたことを示す．学習者のアテンプトが中断されたので，一旦学習アテンプトが再開されると，SCO は前回中断されたアテンプトで得た同一のデータを持つ
3. SCO が cmi.exit を"logout"に設定すると SCO は通常終了となり(SCO の作成者が決定する), SCO の学習アテンプトも通常終了する. SCO の以降の学習アテンプトは新しいランタイムデータの集合を含む.
   ADL Note: "logout"キーワードは反対されており, SCORM の将来のバージョンではサポートされなくなる.
4. SCO が *cmi.exit* を“normal”に設定すると，これは SCO が正常にエグジットしたことを示し，SCO への学習者のアテンプトが正常に終了したことを示す．後続の SCO への学習者のアテンプトは新たなランタイムデータを開始する
5. SCO が *cmi.exit* を“”に設定すると，これはエグジット状態が不明で

	SCO への学習者のアテンプトが終了したことを示す．後続の SCO への学習者のアテンプトは新たなランタイムデータを開始する 注意：どのようにもしくはいつ *cmi.entry* が LMS に設定されるかは実装次第であるが，この値は学習者のアテンプトの中の次の学習セッション中に利用可能になる． **例:** • SetValue(“cmi.exit”,”suspend”).

4.2.9. Interactions

cmi.interactions データモデル要素は，SCO から LMS へ交信される学習者レスポンスを定義する．インタラクションは SCO 開発者が記録したいと思う個々の質問もしくはタスクへのレスポンスとなるように意図されている．相互作用が設定されるように要求されるとき，データの保存以外に意図された LMS の動作はない．

インタラクションは，情報の集合と考えられる．インタラクションデータは，図 4.2.9a に示されている．IEEE ドラフト規格で定義されたように，LMS は，最低 250 セットのインタラクションデータをサポート（保存）するように要求されている[1]．LMS は，250 以上のサポートを提供することを選択することができるが，必要なのは，最低 250 セットのインタラクションデータをサポートする事である．

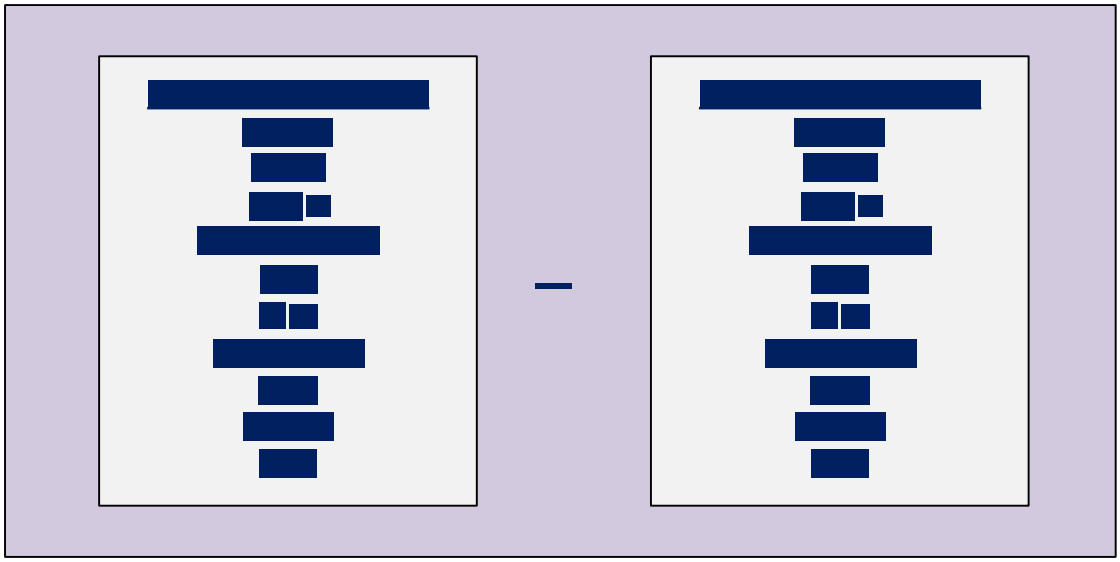

図 4.2.9a: インタラクションおよびインタラクションデータ

インタラクションデータに要求される重要なデータが，識別子(cmi.interactions.n.id)とインタラクションのタイプ(cmi.interactions.n.type)の 二つある．cmi.interactions.n.type データモデル要素が要求されるのは，インタラクションが，正答もしくは学習者の回答に関するデータを含む場合である．これらの二つの情報は，インタラクションの集合の中にある他のインタラクションデータから，そのインタラクションデータを識別する情報である．識別子は，一つのインタラクションを他から固有に指定する．タイプは，インタラクションのタイプ（true-false，マッチングなど）を一意に指定する．

cmi.interactions データモデル要素は，SCO によって，二つの主な手段，journaling と status に対して用いられる：

ジャーナリングスキームは，SCO が，インタラクションデータを，学習者がインタラクションに関わる度に記録することを要求する(新しいインタラクションデータが，インタラクションの配列に追加される)．インタラクションの記録にこのスキームを適用することにより，SCO 内のインタラクションに関する学習者経験を検討するために，情報が集められる．例えば，学習者が，インタラクションに何回レスポンスしたか，各インタラクションに対する待ち時間は何か，学習者の回答は何か，そしてその回答の結果は，等を記述したレポートが生成される．このデータは，将来の利用に向けてインタラクションを潜在的に更新するために集められ用いられる．LMS はこのタイプの情報を独立した研究や調査のために提供するかもしれない．しかし，このデータの利用可能性や特徴は SCORM 規格の範囲外である．

ステータススキームは，SCO にインタラクションデータを記録し，SCO に関する学習者経験に基づいてインタラクションデータを更新することを要求する．例えば，学習者が，インタラクションにレスポンスすると，情報が設定される．学習者がレスポンスを修正すると，インタラクションデータは修正を反映して更新される（インタラクションアレイに新規のエントリーが追加されるのではない）．このスキームでは，インタラクションデータは最後に記録されたインタラクションの状態を含む．このスキームでは，学習者が何回レスポンスを更新したかを追跡する能力は失われる．

どちらかのスキームをもう一方より優先して利用することに対しては，賛否ある．SCO 開発者は二つのスキームがあることを認識し自分の望む方を利用すべきである．LMS の観点からは，ジャーナリングスキームは，保存要件により負担をかける．しかし，保存するインタラクションデータ量の要件は，250 に過ぎない．SCO 開発者は，SPM の 250 が示すのは，LMS が 250 セットのインタラクションデータのサポートを要求しているだけであることを理解する必要がある．

配列に格納された他のデータモデル要素と同じように，インデックスポジション（n）は，格納されたデータの固有のデータセットではない．規格で定義された実装要件は，配列はバッグとして実装されるべきであるとしている[12]．このデータ構造は，同一オブジェクト（インタラクションデータ）が配列でリピートされることを可能にする（セット内のアイテムが固有であることを要求するセットとは異なる）．ステータススキームを用いる場合は SCO が構築されることが推奨され，インタラクションデータモデル要素を用いる場合は固有性に関してインデックスポジションに頼らないことが推奨される．学習者および学習セッションによって，同一のインタラクションデータは，配列の同一のポジションに保存されないかもしれない．SCO は，インタラクションデータを更新する（上記のステータススキーム）前に，相互作用データを通して，指定の識別子を探索するように構築されることが強く推奨される．上記のジャーナリングスキームを利用する場合，学習者がインタラクションをする度に配列で新規のエントリーが作成されるので，この推奨に従う必要はない．

表 4.2.9a: interactions データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.interactions._children	*cmi.interactions._children* データモデル要素は，サポートされたデータモデル要素のリストを表す．このデータモデル要素は SCO によって一般的にどのデータモデル要素が LMS にサポートされているか指定するのに利用される．返却された文字列は，SCO が GetValue()および SetValue()要求に対して動的にパラメータを作成するのに用いられることがある **データモデル要素実装要件:** • ***データタイプ*: 文字列** • **値空間:** ISO-10646-1 [5] • **フォーマット:** LMS にサポートされた親インタラクションデータモデル要素の全てのデータモデル要素のコンマで区切られたリスト．全てのデータモデル要素は LMS にサポートされるように要求されているので，文字列は以下のデータモデル要素を表す： o id o type o objectives o timestamp o correct_responses o weighting o learner_response o result o latency o description

	LMS 動作要件: • このデータモデル要素は必須で，LMS は read-only として実装しなくてはならない • LMS はコンマで区切られた全てのデータモデル要素のリストを返却することに関与する (上記データモデル要素実装要件参照) **SCO 動作要件:** • LMS はこのデータモデル要素を read-only.として実装する必要がある．SCO は *cmi.interactions._children* データモデル要素の値を読み出すことだけが許されている **API 実装要件:** • **GetValue():** LMS はコンマで区切られた LMS にサポートされるデータモデル要素のリストを返却し(上記データモデル要素実装要件参照)，エラーコードを“0” – No error に設定する．データモデル要素の順序は重要でない．返却された文字列はデータモデル要素実装要件で指定された要件を遵守しなくてはならない • **SetValue():** *cmi.interactions._children* を設定するために，SCO が SetValue()要求を呼び出せば，LMS はエラーコードを“404” – Data Model Element Is Read Only に設定し “false” を返却しなくてはならない **例:** • GetValue(“cmi.interactions._children”)
cmi.interactions._count	*cmi.interactions._count* keyword は LMS に保存されているインタラクションの現在の数を記述する．LMS に現在保存されているエントリーの合計数が返却される **データモデル要素実装要件:** • **データタイプ**: 整数 • **値空間**: 負でない整数 • **フォーマット**: LMS が現在保持するインタラクションの数を表す文字列 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS が，インタラクションデータが設定される前に，*cmi.interactions._count* 値を得る要求を受け取れば，LMS は下記の API 実装要件を遵守しなくてはならない **SCO 動作要件:** • LMS はこのデータモデル要素を read-only.として実装する必要がある．SCO は *cmi.interactions._count* データモデル要素の値を読み出すことだけが許されている **API 実装要件:** • **GetValue():** LMS は LMS に現在保存されるインタラクションの数を返却しエラーコードを“0” – No error に設定する o SCO に対してインタラクションデータが利用可能になるまで，LMS は“0”を返却しなくてはならない．これは現在保存されているインタラクションデータがないことを意味する • **SetValue():** *cmi.interactions._count* を設定するために，SCO が SetValue()要求を呼び出せば，LMS はエラーコードを“404” – Data Model Element Is Read Only に設定し “false”を

	返却しなくてはならない **例:** • GetValue("cmi.initeractions._count")
cmi.interactions.n.id	*cmi.interactions.n.id* データモデル要素は，インタラクションに対するラベルである．このラベルは，最低限の SCO のインタラクションレコード毎にユニークである[1] **データモデル要素実装要件:** • **データタイプ:** long_identifier_type • **値空間:** RFC 2396 [6]に基づく有効 URI を表す文字列(SPM: 4000)．URI は RFC 2141 [3]に基づく URN であることが推奨される • **フォーマット:** より詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • LMS はこの API 実装要件で記述される追加要件にそったデータモデル要素の書き込みと読み出しだけに責任がある **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.interactions.n.id* データモデル要素の値を読み出しおよび書き込むことが許されている • SCO はインタラクションデータをシーケンシャル順に設定する責任がある．全ての配列は密な配列として実装される．密な配列はアレイポジションがスキップされる事なく実装される • *cmi.interactions.n.id* は，SCO に追跡される必要がある各インタラクションに対して最初に設定されるように要求される • SCO は，学習者のアテンプト中に，既存のインタラクション ID を変えないことが推奨される．SCO が学習者のアテンプト中に *cmi.interactions.n.id* を変更すると，前の学習セッションで収集されたインタラクションデータが損なわれることがあり，一貫性のないインタラクションに関する情報が提供されることになる **API 実装要件:** • **GetValue():** LMS は，関連する現在 LMS に保存された *cmi.interactions.n.id* を返却し，エラーコードを"0" – No error に設定する．返された文字列はデータモデル要素実装要件で指定された要件を遵守しなくてはならない o 集合に対するデータモデルバインディングは密な配列として表される．LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば(例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを"301" – General Get Failure に設定し，空の文字列("")を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 • **SetValue():** LMS は *cmi.interactions.n.id* を SetValue()要求で供給された値に設定し，エラーコードを"0" – No error に設定し"true"を返却しなくてはならない o SetValue()の供給された値がデータモデル要素実

	装要件の要件を満たさない場合は，LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返却しなくてはならない．LMS はこの要求にもとづいてデータモデル要素の状態を変えてはならない o 集合に対するデータモデルバインディングは密な配列として表される．SCO が現在保持されているインタラクションの数より大きいインデックス(n)を SetValue()要求で呼び出せば，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 **追加動作要件:** SCO は新しいインタラクション情報がインデックスリストに順番に挿入されることを確実にする責任がある．インタラクションの識別子は，インタラクションデータが SCO に追跡されるように要求される場合の、SCO に設定される必須項目である． **例:** • GetValue(“cmi.interactions.0.id”) • SetValue(“cmi.interactions.0.id”,”obj1”)
cmi.interactions.n.type	*cmi.interactions.n.type* データモデル要素は，どのタイプのインタラクションがインタラクションのインスタンスとして記録されたかを示す．また，どのようにインタラクションレスポンスが解釈されるかを決定するのにも利用される[1] **データモデル要素実装要件:** • **データタイプ**: 状態 (true_false, multiple_choice, fill_in, long_fill_in, matching, performance, sequencing, likert, numeric, other) • **値空間**: IEEE ドラフトは 10 個の状態値を定義する．SCORM はこれらの状態値を以下の限定された語彙トークンに定める： o “true-false”: インタラクションは二つの可能なレスポンスを持つ[1]．SCORM は，二つの可能なレスポンスが“true”もしくは “false”であることを要求する．省略されたバージョンや代替型(t,f,1,0 など)は許されていない o “choice”: インタラクションは学習者が選択できる二つ以上の利用可能なレスポンスを持つ[1] o “fill-in”: インタラクションは　学習者に一文字以上の文字列で短いレスポンスを返却することを要求する．一般的に，正答は単語の 1 部，1 単語もしくは数単語から成り立つ[1] o “long-fill-in”: インタラクションは，学習者が長い列のキャラクターの形でレスポンスを供給することを要求する o “likert”: インタラクションは，学習者にスケール上の不連続な選択肢から選択することを要求する[1] o “matching”: インタラクションは 2 セットのアイテムから成り立つ．最初のセットのメンバーは第 2 のセットのゼロ以上のメンバーと関連する o “performance”: インタラクションは，学習者に複数のステップを要求するタスクを実施することを要求する[1]

	○ “sequencing”: インタラクションは学習者がリストの項目を論理的な順序にすることを要求する[1] ○ “numeric”: インタラクションは学習者から数字のレスポンスを要求する．このレスポンスはオプションで小数点が付くことがある単純な数字である[1] ○ “other”: IEEE ドラフト規格で定義されていない他のタイプのインタラクション[1] • **フォーマット:** このデータモデル値のフォーマットは上記のトークンの一つである(“true-false”, “choice”, “fill-in”, “long-fill-in”, “matching”, “performance”, “sequencing”, “likert”, “numeric” or “other”) **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • LMS はこのデータモデル要素の書き込みと読み出し API 実装要件で記述される追加要件だけに責任がある **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.interactions.n.type* データモデル要素の値を読み出しおよび書き込むことが許されている • SCO はインタラクションデータをシーケンシャル順に設定する責任がある．全ての配列は密な配列として実装される．密な配列はアレイポジションがスキップされる事なく実装される • *cmi.interactions.n.type* は，correct_response および/もしくは learner_response が追跡される SCO によって，追跡されなくてはならない各インタラクションに対して設定されるように要求される．*cmi.interactions.n.type* は correct_response もしくは learner_response が設定される前に設定される．*cmi.interactions.n.type* が correct_response もしくは learner_response の前に設定されたないと，データモデル依存関係が満たされない • SCO は，学習者のアテンプト中に，既存のインタラクションタイプを変えないことが推奨される．SCO が学習者のアテンプト中にインタラクションタイプを変更すると，前の学習セッションで収集されたインタラクションデータが損なわれることがある．インタラクションタイプを変更することにより，correct_response および learner_response データが無効になることがある **API 実装要件:** • **GetValue():** LMS は，現在 LMS に保存された *cmi.interactions.n.type* に関連する値を返却し，エラーコードを“0” – No error に設定する．返却された文字列はデータモデル要素実装要件で指定された要件を遵守しなくてはならない ○ 集合に対するデータモデルバインディングは密な配列として表される．LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば (例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照

	○ SCO が *cmi.interactions.n.type* を読み出そうとし，データのレコードが作成されるが，タイプデータモデル要素が SCO に設定されていないと，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返却しなくてはならない • **SetValue():** LMS は *cmi.interactions.n.id* を SetValue()要求で供給された値に設定し，エラーコードを“0” – No error に設定し“true”を返却しなくてはならない ○ *cmi.interactions.n.type* を設定するのに SCO が要求を呼び出し，値が上記の制限された語彙トークンの一つでないなら，LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返却しなくてはならない．LMS はこの要求にもとづていデータモデル要素の状態を変えてはならない ○ 集合に対するデータモデルバインディングは密な配列として表される．SCO が現在保持されているインタラクションの数より大きいインデックス(n)を SetValue()要求で呼び出せば，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 ○ *cmi.interactions.n.id* が *cmi.interactions.n.type* の設定要求よりも前に設定されていないなら, LMS はエラーコードを 408 - Data Model Dependency Not Established に設定して"false"を返し, 現在のデータモデル要素の状態を変化させない. **例:** • GetValue(“cmi.interactions.0.type”) • SetValue(“cmi.interactions.0.type”,”true-false”)
Objectives	
cmi.interactions.n.objectives._count	*cmi.interactions.n.objectives._count* keyword は LMS に現在保存されている目標の数を記述する．LMS に現在保存されているエントリーの合計数が返却される **データモデル要素実装要件:** • **データタイプ**: 整数 • **値空間**: 負でない整数 • **フォーマット**: LMS が保持する目標識別子の数を表す文字列．より詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS が，インタラクション目標識別子が設定される前に，*cmi.interactions.n.objectives._count* 値への要求を受け取れば，LMS は下記の API 実装要件を遵守しなくてはならない **SCO 動作要件:** • LMS はこのデータモデル要素を read-only.として実装する必要がある．SCO は *cmi.interactions.n.objectives._count* デー

	タモデル要素の値を読み出すことだけが許されている **API 実装要件:** • **GetValue():** LMS は LMS に現在保存されるインタラクションに対する目標識別子の数を返却しエラーコードを“0” – No error に設定する o 目標識別子が SCO に保存されるように要求されるまで，LMS は“0”を返却しなくてはならない．これは現在保存されているインタラクションに対する目標識別子がないことを意味する．目標識別子が SCO に保存されるように要求されなければ，LMS は“0”を返却しなくてはならない • **SetValue():** *cmi.interactions.n.objectives._count* を設定するために，SCO が SetValue()要求を呼び出せば，LMS はエラーコードを“404” – Data Model Element Is Read Only に設定し“false”を返却しなくてはならない **例:** • GetValue(“cmi.interactions.0.objectives._count”)
cmi.interactions.n.objectives.m.id	*cmi.interactions.n.objectives.m.id* データモデル要素は，インタラクションと関連する学習目標のラベルである．このラベルは最低 SCO の範囲内でユニークである[1] 目的識別子は目標データモデル要素(*cmi.objectives.n.id*)で見つかる目標識別子と対応することもあればないこともある．関係の有無は実装次第である．SCO はこの情報と関係を追跡するよう設計されることがある *cmi.interactions.n.objectives.m.id* は目標識別子の配列である．LMS は最低 10（要求される SPM）の目標識別子を保持する．LMS はそれ以上保存するようにできるが，これは実装次第である **データモデル要素実装要件:** • **データタイプ:** long_identifier_type • **値空間:** RFC 2396 [6]に基づく有効 URI を表す文字列(SPM: 4000)．URI は RFC 2141 [3]に基づく URN であることが推奨される • **フォーマット:** より詳細な long_identifier_type のフォーマットの要件の情報はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • LMS はこのデータモデル要素の書き込みと読み出しおよび API 実装要件で記述される追加要件だけに責任がある • LMS は最低 10 の目標識別子を保存できるようにする（SPM 要件） **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.interactions.n.objectives.m.id* データモデル要素の値を読み出しおよび書き込むことが許されている • SCO はインタラクションデータをシーケンシャル順に設定する責任がある．全ての配列は密な配列として実装される．密な配列はアレイポジションがスキップされる事なく実装される • SCO は関連するいづれの目標識別子も設定する前に

	cmi.interactions.n.id を設定していることを要求される **API 実装要件:** • **GetValue():** LMS は，現在 LMS に保存された *cmi.interactions.n.objectives.m.id* に関連する値を返却し，エラーコードを“0” – No error に設定する．返却された文字列はデータモデル要素実装要件で指定された要件を遵守しなくてはならない o 集合に対するデータモデルバインディングは密な配列として表される．S LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば (例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 • **SetValue():** LMS は *cmi.interactions.n.objectives.m.id* を SetValue()要求で供給された値に設定し，エラーコードを“0” – No error に設定し“true”を返却しなくてはならない o SetValue()の供給された値がデータモデル要素実装要件の要件を満たさない場合は，LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返却しなくてはならない．LMS はこの要求にもとづいていデータモデル要素の状態を変えてはならない o 集合に対するデータモデルバインディングは密な配列として表される．SCO が，保持されている目標の現在の数(*cmi.interactions.n.objectives._count の要求*で決められる)もしくは現在保持されているインタラクションの数(*cmi.interactions._count の要求*で決められる)より大きいインデックス(n)を SetValue()要求で呼び出せば，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o *cmi.interactions.n.objectives.m.id* を設定する要求の前に *cmi.interactions.n.id* が設定されていない場合は，LMS はエラーコードを“408” – Data Model Dependency Not Established に設定し“false”を返却し，データモデル要素の現在の状態を変えてはならない o SetValue()の供給された値が学習アテンプトのより早い配列上の位置で既に使われた値(インタラクション学習目標識別子のセットの範囲内で一意でない)であるなら, LMS はエラーコードを 351 - General Set Failure に設定し"false"を返す. セクション 3.1.7.6: *SCORM Extension Error Conditions* を参照のこと. LMS は要求に基づくデータモデル要素の状態を変えない. **例:** • GetValue(“cmi.interactions.0.objectives.0.id”) • SetValue(“cmi.interactions.0.objectives.0.id”,“urn:ADL:objective-id-0001”)
cmi.interactions.n.timestamp	*cmi.interactions.n.timestamp* データモデル要素はインタラクションを

	始めた学習者の学習インタラクションおよび回答が利用可能になった時点である [1]．タイムスタンプの値は時間の 1 点で表される．いくつかのインタラクションが同時に提示されたら，これらは同一のタイムスタンプ値を持つ．インタラクションがレスポンスに利用可能にならない場合（適用的テストには利用されないインタラクションなど），このインタラクションにはタイムスタンプ値は利用されない．タイムスタンプ値がインタラクションに利用可能で学習者レスポンスデータが利用可能でない場合，インタラクションは学習者に利用可能であるが，学習者がインタラクションにレスポンスしなかったと仮定される． **データモデル要素実装要件:** • **データタイプ**: 時間(秒,10,0) • **値空間**: このデータタイプは時間に対する値が，1 秒およびオプションの 100 分の 1 秒の精度で実数データタイプとして表される数字であることを示す．時間値の秒の数字は 1970 年 1 月 1 日 00:00 からのものである • **フォーマット**: 時間(秒,10,0)データタイプの必要フォーマットの情報の詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • LMS はこのデータモデル要素の保管および読み出しおよび API 実装要件で記述される追加要件だけに責任がある **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.interactions.n.timestamp* データモデル要素値を読み出しおよび書き込むことが許されている • SCO はインタラクションデータをシーケンシャル順に設定する責任がある．全ての配列は密な配列として実装される．密な配列はアレイポジションがスキップされる事なく実装される • SCO はどんな関連タイムスタンプを設定するときも前に *cmi.interactions.n.id* を設定していることが要求される **API 実装要件:** • **GetValue()**: LMS は，関連する現在 LMS に保存されたインタラクションタイムスタンプを返却し，エラーコードを “0” – No error に設定する．返却された文字列はデータモデル要素実装要件で指定された要件を遵守しなくてはならない o 集合に対するデータモデルバインディングは密な配列として表される．LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば (例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返却しなくてはならない．この処理に関するより進んだ提言の詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o SCO が *cmi.interactions.n.timestamp* を読み出そうとして，データのレコードが作成されるが，*cmi.interactions.n.timestamp* が SCO に設定されていない場合は，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文

	字列(“”)を返却しなくてはならない • **SetValue():** LMS は *cmi.interactions.n.timestamp* を SetValue() 要求で供給された値に設定し，エラーコードを“0” – No error に設定し“true”を返却しなくてはならない o SetValue()の供給された値がデータモデル要素実装要件の要件を満たさない場合は，LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返却しなくてはならない．LMS はこの要求にもとづいていデータモデル要素の状態を変えてはならない o 集合に対するデータモデルバインディングは密な配列として表される．SCO が，現在保持されているインタラクションの数より大きいインデックス(n)を SetValue()要求で呼び出せば，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o *cmi.interactions.n.timestamp* を設定する要求の前に *cmi.interactions.n.id* が設定されていない場合は，LMS はエラーコードを“408” – Data Model Dependency Not Established に設定し“false”を返却し，データモデル要素の現在の状態を変えてはならない **例:** • GetValue(“cmi.interactions.0.timestamp”) • SetValue(“cmi.interactions.0.timestamp“2003-07-25T03:00:00”)
correct_responses	
cmi.interactions.n.correct_responses._count	*cmi.interactions.n.correct_responses._count* keyword は，LMS に現在保存された正答の数を記述するのに利用される．LMS に現在保存されたエントリーの合計数が返却される **データモデル要素実装仕様:** • **データタイプ**: 整数 • **値空間**: 負でない整数 • **フォーマット**: LMS が保持する正答の数を表す文字列．整数のデータタイプのフォーマット要件のより多くの情報の詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS が，インタラクションに対する正答が設定される前に，*cmi.interactions.n.correct_responses._count* 値を得る要求を受け取れば，LMS は下記の API 実装要件を遵守しなくてはならない **SCO 動作要件:** • LMS はこのデータモデル要素を read-only.として実装する必要がある．SCO は *cmi.interacatins.n.correct_responses._count* データモデル要素の値を読み出すことだけが許されている **API 実装要件:** • **GetValue():** LMS は LMS に現在保存されるインタラクションに対する正答の数を返却しエラーコードを“0” – No error に設定する

	○ 正答が SCO に保存されるように要求されるまで，LMS は“0”を返却しなくてはならない．これは現在保存されているインタラクションに対する正答がないことを意味する • **SetValue():** *cmi.interactions.n.correct_responses._count* を設定するために，SCO が SetValue()要求を呼び出せば，LMS はエラーコードを“404” – Data Model Element Is Read Only に設定し “false”を返却しなくてはならない **例:** • GetValue(“cmi.interactions.0.correct_responses._count”)
cmi.interactions.n.correct_responses.n.pattern	*cmi.interactions.n.correct_responses.n.pattern* データモデル要素は，インタラクションに対する一つの正しいレスポンスパターンを定義する．パターン値のフォーマットはインタラクションのタイプ(*cmi.interactions.n.type*)による． *cmi.interactions.n.correct_responses* の集合はインタラクションに対する正しいレスポンスの密な配列である．インタラクションのタイプ(*cmi.interactions.n.type*)により，サポートに要求される正しいレスポンスパターンの数は異なる．それぞれのタイプのより詳細はセクション 4.2.9.1: *正解パターンデータモデル要素詳細*参照 **データモデル要素実装要件:** • パターンに対するデータタイプおよびフォーマットはインタラクションのタイプ(*cmi.interactions.n.type*)によって異なる．セクション 4.2.9.1: *Correct Responses Pattern Data Model Element Specifics* は各タイプのインタラクションに対するデータモデル要素実装要件およびフォーマットを定義する • SetValue()および GetValue()処理中，LMS および SCO は，インタラクションタイプに基づくパターンのフォーマットに気をつける必要がある．どのように LMS および/もしくは SCO がパターンを処理するかは SCORM の対象範囲外である．例えば，SCO は，学習者に値を表示するのに，LMS から返却された文字列を解析する必要があることがある **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO はもし適切であるならこの値を設定する責任がある．LMS はインタラクションに対して何が正答か判断を下せない．LMS が SCO に値を設定される前に GetValue()要求を受け取れば，LMS は適切なエラーコードを設定する (API 実装要件参照) **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.interactions.n.correct_responses.n.pattern* データモデル要素の値を読み出しおよび書き込むことが許されている．正答に対するデータモデル要素要件およびフォーマットはセクション 4.2.9.1: *Correct Responses Pattern Data Model Element Specifics* に詳細が記述されている • 関連するインタラクションの正答を設定する前に，SCO は *cmi.interactions.n.id* および *cmi.interactions.n.type*

	cmi.interactions.n.id データモデル要素を設定していることを要求される **API 実装要件:** • **GetValue():** LMS は，関連する現在 LMS に保存された *cmi.interactions.n.correct_responses.n.pattern* を返却し，エラーコードを“0” – No error に設定する．返却された文字列はデータモデル要素実装要件で指定された要件を遵守しなくてはならない o 集合に対するデータモデルバインディングは密な配列として表される．LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば(例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返却しなくてはならない．この処理に関するより進んだ提言の詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 • **SetValue():** LMS は *cmi.interactions.n.correct_responses.n.pattern* データモデル要素を SetValue()コールの parameter_2 として通ったパレメータへ設定し，エラーコードを“0” - No error に設定し“true”を返却しなくてはならない o *cmi.interactions.n.correct_responses.n.pattern* を設定する要求の前に *cmi.interactions.n.id* もしくは *cmi.interactions.n.type* が設定されていない場合は，LMS はエラーコードを“408” – Data Model Dependency Not Established に設定し“false”を返し，データモデル要素の現在の状態を変えてはならない o SCO が *cmi.interactions.n.correct_responses.n.pattern* をセクション 4.2.9.1: *Correct Responses Pattern Data Model Element Specifics* で定義された要件に満たない値に設定する場合は，LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返却しなくてはならない．LMS はこの要求にもとづいていデータモデル要素の状態を変えてはならない 集合に対するデータモデルバインディングは密な配列として表される．SCO が現在保持されているインタラクションの数，もしくは正しいレスポンスパターンより大きいインデックス(n)を SetValue()要求で呼び出せば，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 **例:** • GetValue(“cmi.interactions.0.correct_responses.1.pattern”) • SetValue(“cmi.interactions.0.correct_responses.0.pattern”,”true”)
cmi.interactions.n.weighting	*cmi.interactions.n.weighting* データモデル要素は，スコアに対する値を計算するために利用されるインタラクションに与えられるウエイトである．インタラクションウエイトは一般的にスコアの値に対するインタラクションの影響を説明するのに利用されるが，SCO 以外のシステムにスコアを計算するために利用されることを意図されていない [1]．どのようにこの値もしくは合計スコアの計算が SCO に

	よって計算されるかは SCORM の対象範囲外である **データモデル要素実装要件:** • **データタイプ:** 実数(10,7) • **値空間:** 有効桁数 7 桁の精度を持つ実数 • **フォーマット:** 実数(10,7)の要件フォーマットのさらなる詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO はもし適切であるならこの値を設定する責任がある．LMS はインタラクションウエイトには判断を下せない．LMS が SCO に値を設定される前に GetValue()要求を受け取れば，LMS は適切なエラーコードを設定する (API 実装要件参照) **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.interactions.n.weighting* データモデル要素の値を読み出しおよび書き込むことが許されている • 関連するインタラクションウエイトを設定する前に，SCO が *cmi.interactions.n.id* データモデル要素を設定していることを要求される **API 実装要件:** • **GetValue():** LMS は，関連する現在 LMS に保存された *cmi.interactions.n.weighting* を返却し，エラーコードを“0” – No error に設定する．返却された文字列はデータモデル要素実装要件で指定された要件を遵守しなくてはならない o 集合に対するデータモデルバインディングは密な配列として表される．LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば (例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返却しなくてはならない．この処理に関するより進んだ提言の詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o SCO は *cmi.interactions.n.weighting* を読み出そうとし，データのレコードが作成されるが，*cmi.interactions.n.weighting* が SCO に設定されていない場合は，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返却しなくてはならない • **SetValue():** LMS は *cmi.interactions.n.weighting* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータに設定し，エラーコードを“0” – No error に設定し“true”を返却しなくてはならない o SCO が *cmi.interactions.n.weighting* を実数でない値に設定しようとすると，LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返却しなくてはならない．LMS はこの要求にもとづいていデータモデル要素の状態を変えてはならない o 集合に対するデータモデルバインディングは密な

	配列として表される．SCO が，現在保持されているインタラクションの数(*cmi.interactions._count の要求*で決められる)より大きいインデックス(n)を SetValue()要求で呼び出せば，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o *cmi.interactions.n.weighting* を設定する要求の前に *cmi.interactions.n.id* が設定されていないと，LMS はエラーコードを“408” – Data Model Dependency Not Established に設定し“false”を返却し，データモデル要素の現在の状態を変えてはならない **例:** • GetValue(“cmi.interactions.0.weighting”) • SetValue(“cmi.interactions.0.weighting”,”1.0”)
cmi.interactions.n.learner_response	*cmi.interactions.n.learner_response* データモデル要素は，学習者がインタラクションにレスポンスしたときに生成されるデータから成り立つ [1]．学習者レスポンスは 10 個の利用可能なバリアントのうち一つを持つ．各バリアントはインタラクションのタイプ (*cmi.interactions.n.type*)による．より詳細はセクション 4.2.9.2: *学習者レスポンスデータモデル要素詳細*参照 **データモデル要素実装要件:** • 値に対するデータタイプはインタラクションのタイプ (*cmi.interactions.n.type*)によって異なる．セクション 4.2.9.2: *Learner Response Data Model Element Specifics* は各インタラクションタイプのデータモデル要素実装要件およびフォーマットを定義する • SetValue()および GetValue()処理中，LMS および SCO は，学習者レスポンスのフォーマットがインタラクションタイプに基づくことに注意する必要がある．どのように LMS および/もしくは SCO が学習者レスポンスを処理するかは SCORM の対象範囲外である．例えば，SCO は，学習者に値を表示するのに，LMS から返却された文字列を解析する必要があることがある **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO はもし適切であるならこの値を設定する責任がある．LMS はインタラクションに対して学習者レスポンスが正しいかどうか判断を下せない．LMS が SCO に値を設定する前に GetValue()要求を受け取れば，LMS は適切なエラーコードを設定する (API 実装要件参照) **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.interactions.n.learner_response* データモデル要素の値を読み出しおよび書き込むことが許されている．学習者レスポンスに対するデータモデル要素要件およびフォーマットはセクション 4.2.9. 2: *Learner Response Data Model Element Specifics* に詳細が記述されている • 関連するインタラクション学習者レスポンスを設定する前に，SCO は *cmi.interactions.n.id* および *cmi.interactions.n.type*

	データモデル要素を設定していることを要求される • *cmi.interactions.n.learner_response* は設定の要求をされない. 学習者がインタラクションに応答しなければ, *cmi.interactions.n.learner_response* が設定されるかを決めるのは SCO である. 学習者がインタラクションに応答しないならば *cmi.interactions.n.learner_response* を設定しないことは最善の策だろう. **API 実装要件:** • **GetValue():** LMS は，関連する現在 LMS に保存された *cmi.interactions.n.learner_response* を返却し，エラーコードを“0” – No error に設定する．返却された文字列はデータモデル要素実装要件で指定された要件を遵守しなくてはならない o 集合に対するデータモデルバインディングは密な配列として表される．LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば(例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o SCO が *cmi.interactions.n.learner_response* を読み出そうとし，データのレコードが作成されるが，learner_response データモデル要素が SCO に設定されていないと，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返却しなくてはならない • **SetValue():** LMS は *cmi.interactions.n.learner_response* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータに設定し，エラーコードを“0” – No error に設定し“true”を返却しなくてはならない o SCO が *cmi.interactions.n.learner_response* をセクション 4.2.9.2: *Learner Response Data Model Element Specifics* で定義された要件に満たない値に設定しようとすると，LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返却しなくてはならない．LMS はこの要求にもとづていデータモデル要素の状態を変えてはならない o 集合に対するデータモデルバインディングは密な配列として表される．SCO が，現在保持されているインタラクションの数より大きいインデックス(n)を SetValue()要求で呼び出せば，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o *cmi.interactions.n.learner_response* を設定する要求の前に，*cmi.interactions.n.id* および *cmi.interactions.n.type* が設定されていない場合は，LMS はエラーコードを“408” – Data Model Dependency Not Established に設定し“false”を返却し，データモデル要素の現在の状態を変えてはならない

	例: • GetValue("cmi.interactions.0.learner_response") • SetValue("cmi.interactions.0.learner_response","true")
cmi.interactions.n.result	*cmi.interactions.n.result* データモデル要素は，学習者のレスポンスに対する正しさの判断である[1] **データモデル要素実装要件:** • **データタイプ**: 状態 (correct, incorrect, unanticipated, neutral, real (10,7)) • **値空間**: IEEE ドラフトは 5 個の状態値を定義する．SCORM はこれらの値を以下の制限された語彙トークンに定める： o "correct": 学習者レスポンスは正しかった[1] o "incorrect": 学習者レスポンスは間違いだった[1] o "unanticipated": 学習者レスポンスは予想されていなかった [1] o "neutral": 学習者レスポンスは正しくも間違いでもなかった[1] o real (10,7): 実数．この結果は，学習者レスポンスの正しさの数字的予測をレポートすることを提供する[1] • **フォーマット**: データモデル値のフォーマットは，上記のトークンの一つである ("correct", "incorrect", "unanticipated", "neutral")もしくは有効桁数 7 桁の実数である **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO はもし適切であるならこの値を設定する責任がある．LMS はインタラクションの学習者レスポンスに対して結果の判断を下せない．LMS が SCO に値を設定される前に GetValue()要求を受け取れば，LMS は適切なエラーコードを設定しなければならない (API 実装要件参照) **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.interacations.n.result* データモデル要素の値を読み出しおよび書き込むことが許されている • 関連するインタラクションの結果を設定する前に，SCO は *cmi.interactions.n.id* データモデル要素を設定していることを要求される **API 実装要件:** • **GetValue()**: LMS は，関連する現在 LMS に保存された *cmi.interactions.n.result* を返却し，エラーコードを"0" – No error に設定する．返却された文字列はデータモデル要素実装要件で指定された要件を遵守しなくてはならない o 集合に対するデータモデルバインディングは密な配列として表される．LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば (例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを"301" – General Get Failure に設定し，空の文字列("")を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照

	○ SCO が *cmi.interactions.n.result* を読み出そうとし，データのレコードが作成されるが，*cmi.interactions.n.result* が SCO に設定されていないと，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返却しなくてはならない • **SetValue():** LMS は *cmi.interactions.n.result* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータに設定し，エラーコードを“0” – No error に設定し“true”を返却しなくてはならない ○ SCO が *cmi.interactions.n.result* をデータモデル実装要件に満たない値に設定しようとすると，LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返却しなくてはならない．LMS はこの要求にもとづいてデータモデル要素の状態を変える事はない ○ 集合に対するデータモデルバインディングは密な配列として表される．SCO が，現在保持されているインタラクションの数より大きいインデックス(n)を SetValue()要求で呼び出せば，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 ○ *cmi.interactions.n.result* を設定する要求の前に *cmi.interactions.n.id* が設定されていない場合は，LMS はエラーコードを“408” – Data Model Dependency Not Established に設定し“false”を返却し，データモデル要素の現在の状態を変えてはならない **例:** • GetValue(“cmi.interactions.0.result”) • SetValue(“cmi.interactions.0.result”,”1.0”) • SetValue(“cmi.interactions.0.result”,”correct”)
cmi.interactions.n.latency	*cmi.interactions.n.latency* データモデル要素は，学習者にレスポンスに対してインタラクションが利用可能になった時間から最初のレスポンスまでの経過時間である．待ち時間情報は，学習者がレスポンスしないとインタラクションから入手できない．待ち時間は，インタラクションの *cmi.interactions.n.timestamp* と最初のレスポンスの時間との時間差である[1] **データモデル要素実装要件:** • **データタイプ:** timeinterval (second,10,2) – 決断するまでの 0.01 秒単位の時間間隔 • **フォーマット:** より詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO はもし適切であるならこの値を設定する責任がある．LMS はインタラクションの待ち時間の判断を下せない．LMS が SCO に値を設定される前に GetValue()要求を受け取れば，LMS は適切なエラーコードを設定する (API 実装

	要件参照) **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.interactions.n.latency* データモデル要素の値を読み出しおよび書き込むことが許されている • 関連するインタラクションの待ち時間を設定する前に，SCO は *cmi.interactions.n.id* データモデル要素を設定していることを要求される **API 実装要件:** • **GetValue():** LMS は，関連する現在 LMS に保存された *cmi.interactions.n.latency* を返却し，エラーコードを“0” – No error に設定する．返却された文字列はデータモデル要素実装要件で指定された要件を遵守しなくてはならない o 集合に対するデータモデルバインディングは密な配列として表される．LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば (例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o SCO が *cmi.interactions.n.result* を読み出そうとし，データのレコードが作成されるが，*cmi.interactions.n.latency* が SCO に設定されていないと，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返却しなくてはならない • **SetValue():** LMS は *cmi.interactions.n.latency* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータに設定し，エラーコードを“0” – No error に設定し“true”を返却しなくてはならない o SCO が *cmi.interactions.n.latency* をデータモデル実装要件に満たない値に設定しようとすると，LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返却しなくてはならない．LMS はこの要求にもとづいてデータモデル要素の状態を変えてはならない o 集合に対するデータモデルバインディングは密な配列として表される．SCO が，現在保持されているインタラクションの数より大きいインデックス(n)を SetValue()要求で呼び出せば，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o *cmi.interactions.n.latency* を設定する要求の前に *cmi.interactions.n.id* が設定されていないと，LMS はエラーコードを“408” – Data Model Dependency Not Established に設定し“false”を返却し，データモデル要素の現在の状態を変えてはならない **例:** • GetValue(“cmi.interactions.0.latency”) • SetValue(“cmi.interactions.0.latency”,” PT5M”) – 5 分間

cmi.interactions.n.description	*cmi.interactions.n.description* データモデル要素はインタラクションの短い情報目的の記述である **データモデル要素実装要件:** • **データタイプ:** localized_string_type (SPM: 250) • **値空間: ローカライズされた文字列を表す文字列** • **フォーマット: より詳細はセクション** 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO はもし適切であるならこの値を設定する責任がある．LMS はインタラクションに対して，SCO からレポートされていない限り，記述の判断を下せない．LMS が SCO に値を設定される前に GetValue()要求を受け取れば，LMS は適切なエラーコードを設定する (API 実装要件参照) **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.interactions.n.description* データモデル要素の値を読み出しおよび書き込むことが許されている • 関連するインタラクション記述を設定する前に，SCO は *cmi.interactions.n.id* データモデル要素を設定していることを要求される • SetValue()要求中，SCO は，{lang=<language_type>}デリミターはオプションであることに注意しなくてはならない．デリミターが文字列の 1 部でなければ，LMS はデフォルト言語は “en”（英語）であると仮定する • GetValue()要求中，SCO は，{lang=<language_type>}デリミターは LMS に返却される文字列の 1 部である(LMS の実装による)ことがあることに注意しなくてはならない．SCO が LMS に返却された文字列と何をするかは SCO の実装による．文字列がデリミターを含まなければ，SCO は言語は “en”（英語）であると仮定する **API 実装要件:** • **GetValue():** LMS は，関連する現在 LMS に保存された *cmi.interactions.n.description* を返却し，エラーコードを“0” – No error に設定する．返却された文字列はデータモデル要素実装要件で指定された要件を遵守しなくてはならない o 集合に対するデータモデルバインディングは密な配列として表される．LMS が現在保持する数よりインデックス（n）が大きい GetValue()要求を SCO が呼び出せば (例えば，要求の n 値は 5 で，配列には 3 コメントしかない)，LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o SCO が *cmi.interactions.n.description* を読み出そうとし，データのレコードが作成されるが，*cmi.interactions.n.description* が SCO に設定されていないと，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返却しなくてはならない

	• **SetValue():** LMS は *cmi.interactions.n.description* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータに設定し，エラーコードを“0” – No error に設定し“true”を返却しなくてはならない o SCO が *cmi.interactions.n.description* をデータモデル実装要件に満たない値に設定しようとすれば，LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返却しなくてはならない．LMS はこの要求にもとづていデータモデル要素の状態を変えてはならない o 集合に対するデータモデルバインディングは密な配列として表される．SCO が，現在保持されているインタラクションの数より大きいインデックス(n)を SetValue()要求で呼び出せば，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返却しなくてはならない．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o *cmi.interactions.n.description* を設定する要求の前に *cmi.interactions.n.id* が設定されていない場合，LMS はエラーコードを“408” – Data Model Dependency Not Established に設定し“false”を返却し，データモデル要素の現在の状態を変えてはならない **例:** • GetValue(“cmi.interactions.0.description”) • SetValue(“cmi.interactions.0.description”,”以下のどちらが赤か？”) - デフォルト言語の en（英語）が利用される

4.2.9.1 Correct Responses Pattern データモデル要素詳細

cmi.interactions.n.correct_responses.n.pattern データモデル要素は相互作用タイプ(cmi.interactions.n.type)によって異なる値を持つ．このセクションは，パターンによって定義される文字列値に対するデータモデル値の要件を定義する．

表 4.2.9.1a: Correct Responses Pattern フォーマット要件

相互作用 Data Type	**正解パターン フォーマット**
true_false	IEEE ドラフトは true_false 相互作用タイプに対する正解を以下の通り定義する： true_false: 状態(true, false) **データモデル要素実装要件:** • **データタイプ**: 文字列 • **値空間**: ISO-10646-1 • **フォーマット**: IEEE ドラフトは二つの状態値を定義する．SCORM はこれらの値を以下の制限された語彙トークンにバインドする： o “true”: 正解は true であることを示す o “false”: 正解は false であることを示す **LMS 動作要件:** • 各正解パターンは文字列によって表させる．文字列は上記の**データモデ**

	ル要素実装要件に準拠し，正解集合のある特定のインデックスに保存される • LMS **はこのタイプの相互作用に対する正解は一つしかない場合に実施する．SCO が，追加のパターンを保存するために**(correct_responses index, *n*, 0 <)，SetValue()**を呼び出そうとすると，LMS はエラーコートを**“351” – General Set Failure **に設定し**“false”**を返す．LMS はパターンの有効リストに新たな値を追加しない，もしくは** LMS **は現在保存されたパターンの状態を変更しない** **例:** • SetValue(“cmi.interactions.0.correct_responses.0.pattern”, “true”)

multiple_choice	IEEE **ドラフトは** multiple_choice **相互作用タイプに対する正解を以下の通り定義する：**
	multiple_choice:
	set of set of short_identifier_type
	// The set SPM: 10 sets
	// The set of short_identifier_type SPM: 36 short_identifier_type
	データモデル要素実装要件:
	• **データタイプ: 文字列**
	• **値空間:** a set of short_identifier_type **を表す** ISO-10646-1 **文字列（セットの各要素はリザーブされたトークン**(“[,]”)**によって区切られる）．** LMS **は最低** 36 short_identifier_types **を含む文字列をサポートする．**LMS **はそれ以上をサポートすることが可能であるが，これは実装次第である．**LMS **が責任があるのは，**SPM **の** 36 short_identifier_types **を管理することだけである．**short_identifier_type **データタイプのフォーマットに対する要件に関するより詳細情報はセクション** 4.1.1.7: *データタイプ***参照**
	• **フォーマット: 文字列のフォーマットは：**
	o <short_identifier_type>[,]<short_identifier_type>
	文字列を作るとき以下の要件が遵守されるべきである：
	• **セットは一つ以上の** short_identifier_type **のセットを含む**
	• short_identifier_type **のセットはゼロ以上の** short_identifier_type **を含む．** short_identifier_type **のセットが空であれば，正解は選択肢がないことを表す**
	• **セットが一つ以上の** short_identifier_type (**相互作用は複数の正解を持つ– 全ての正解が要求される)を含めば，各** short_identifier_type **は特殊なリザーブされたトークン**“[,]” **で区切られる**
	• **各** short_identifier_type **はセットに一度だけ発生する**
	• short_identifier_type **の順序は重要ではない**
	LMS 動作要件:
	• **各正解パターンは文字列によって表させる．文字列は上記の*データモデル要素実装要件*に準拠し，正解集合のある特定のインデックスに保存される**
	• LMS **は，正解集合に対して最低** 10 (**要求される** SPM) **パターンをサポートする．**LMS **はそれ以上をサポートすることが可能であるが，これは実装次第である．**LMS **が責任があるのは，**SPM **の** 10 **を管理することだけである．**
	• LMS **は正解セットのユニーク性を強固に守る．**SCO **が既に正解集合に存在するセットを保存するのに** SetValue()**を呼び出そうとすると，**LMS **はエラーコードを**“351” – General Set Failure **に設定し**“false”**を返す．**
	LMS **はパターンの有効リストに新たな値を追加せず，現在保存されたパターンの状態も変更しない**
	例:

	• SetValue("cmi.interactions.0.correct_responses.0.pattern","choice1[,]choice2[,]choice3") • SetValue("cmi.interactions.0.correct_responses.1.pattern","choice1[,]choice2") choice 相互作用のこの例では正解集合は二つのパターンがある．API メソッドコールは正解が以下のいずれかであることを示す • "choice1", "choice2" and "choice3" もしくは • "choice1" and "choice2"

fill_in	IEEE ドラフトは fill_in 相互作用タイプに対する正解を以下の通り定義する：
	
	fill_in:
	bag of record
	{
	case_matters:
	boolean,
	order_matters:
	boolean,
	match_text:
	array (0..9) of localized_string_type(250),
	// The match_text array SPM: 10 localized_string_types
	// The localized_string_type SPM: 250 characters
	}// The bag of record SPM: 5 records
	
	データモデル要素実装要件:
	• **データタイプ**: 文字列
	• **値空間**: アレイの localized_string_type を表す ISO-10646-1 文字列．アレイの各要素はリザーブされたトークン("[,]")によって区切られる． LMS は最低 10 localized_string_types を含む文字列をサポートする．LMS はそれ以上をサポートすることが可能であるが，これは実装次第である．LMS が責任があるのは，SPM の 10 localized_string_types を管理することだけである．localized_string_type データタイプのフォーマットに対する要件に関するより詳細情報はセクション 4.1.1.7: *データタイプ*参照
	• **フォーマット**: 文字列のフォーマットは以下の通り：
	o {case_matters=<boolean>}{order_matters=<boolean>}<localized_string_type>[,]<localized_string_type>
	
	文字列を作るとき以下の要件が遵守されるべきである：
	o アレイは一つ以上の localized_string_types を含む
	o アレイが一つ以上の localized_string_type (相互作用は複数の正解を持つ– 全ての正解が要求される)を含めば，特殊なリザーブされたトークン"[,]" で区切られる
	o 各 localized_string は一度だけ発生する
	o {case_matters=<*boolean*>}デリミターは case が正解パターンの評価に関係するかどうかを示す．case_matters 値はアレイの全ての localized_string_types に適用する．{case_matters=<*boolean*>}は，{order_matters=<*boolean*>}デリミターの前か後に来ることがある．このデリミターはオプションであり，特定されない場合，case_matters のデフォルト値は"false"である．詳細はセクション 4.1.1.6:*予約されたデリミタ*参照
	o {order_matters=<boolean>}デリミターは order が正解パターンの評価に関係するかどうかを示す．order_matters 値はアレイの全ての localized_string_types に適用する．{order_matters=<*boolean*>}は，{case_matters=<*boolean*>}デリミターの前か後に来ることがある．このデリミターはオプションであり，特定されない場合，order_matters のデフォルト値は "true"である．詳細はセクション 4.1.1.6: *予約されたデリミタ*参照
	
	LMS 動作要件:

	• 各正解パターンは文字列によって表させる．文字列は上記の**データモデル要素実装要件**に準拠し，正解集合のある特定のインデックスに保存される • LMS は，正解集合に対して最低 5 (要求される SPM) パターンをサポートする．LMS はそれ以上をサポートすることが可能であるが，これは実装次第である．LMS が責任があるのは，SPM の 5 を管理することだけである． **例:** • SetValue(“cmi.interactions.0.correct_responses.0.pattern”, “{case_matters=true}{lang=en}car”) • SetValue(“cmi.interactions.0.correct_responses.1.pattern”, “{case_matters=true}{lang=en}automobile” これは正解集合が二つのパターンを持つ fill_in 相互作用の例である．API メソッドコールは，正解は “car” もしくは“automobile”であることを示す．正解は全て小文字でなければならず，localized_string_type の言語は英語である • SetValue(“cmi.interactions.0.correct_responses.0.pattern”,”{lang=en}car”) これは正解集合が 1 つのパターンを持つ fill_in 相互作用の例である．API メソッドコールは，正解は“car”であることを示す．正解のケースは関係なく，localized_string_type の言語は英語である • SetValue(“cmi.interactions.0.correct_responses.0.pattern”,”car”) これは正解集合が 1 つのパターンを持つ fill_in 相互作用の例である．API メソッドコールは，正解は“car”であることを示す．リザーブされたデリミターはどちらも特定されていないのでデフォルト値が利用される．この例では，正解のケースは関係なく，localized_string_type の言語は英語である • SetValue(“cmi.interactions.0.correct_responses.0.pattern”,”{case_matters=true}{order_matters=true}car[,]automobile”) これは正解集合が 1 つのパターンを持つ fill_in 相互作用の例である．API メソッドコールは，正解は“car” および“automobile”でこの順番であることを示す．正解の小文字でなければならず，localized_string_type の言語は英語である

long_fill_in	IEEE ドラフトは long_fill_in 相互作用タイプに対する正解を以下の通り定義する：
	long_fill_in:
	bag of record
	{
	case_matters:
	boolean,
	match_text:
	localized_string_type(4000),
	// The localized_string_type SPM: 4000 characters
	}// The bag SPM: 5 records
	データモデル要素実装要件:
	• **データタイプ**: 文字列
	• **値空間:** localized_string_type を表す ISO-10646-1 文字列． 詳細はセクション 4.1.1.7: *データタイプ*参照
	• **フォーマット**: 文字列のフォーマットは：
	o {case_matters=<boolean>}<localized_string_type>
	文字列を作るとき以下の要件が遵守されるべきである：
	o 文字列は一つの localized_string_type を含まなければならない
	o {case_matters=<*boolean*>}デリミターは case が正解パターンの評価に関係するかどうかを示す．このデリミターはオプションであり，特定されない場合，case_matters のデフォルト値は“false”

	である．詳細はセクション 4.1.1.6: *予約されたデリミタ*参照 **LMS 動作要件:** • 各正解パターンは文字列によって表させる．文字列は上記の**データモデル要素実装要件**に準拠し，正解集合のある特定のインデックスに保存される • LMS は，正解集合に対して最低 5 (要求される SPM) パターンをサポートする．LMS はそれ以上をサポートすることが可能であるが，これは実装次第である．LMS が責任があるのは SPM の 5 を管理することだけである **例:** 下記の例は Gettysburg Address 全体を含まない．スペースをセーブしてデータタイプを記述するためにだけが書かれている．実践的な利用は Gettysburg Address の全テキストを記述することである • SetValue("cmi.interactions.0.correct_responses.0.pattern", "{case_matters=true}{lang=en}Four score and seven years ago…") これは正解集合が 1 つのパターンを持つ long_fill_in 相互作用の例である．API メソッドコールは正解は Gettysburg Address の一部であることを示す．正解のケースは関係なく，localized_string_type の言語は英語である • SetValue("cmi.interactions.0.correct_responses.0.pattern","{lang=en}Four score an seven years ago…") これは正解集合が 1 つのパターンを持つ long_fill_in 相互作用の例である．API メソッドコールは正解は Gettysburg Address の一部であることを示す．{case_matters=<boolean>}デリミターが特定されないので，デフォルト値（False）が利用される．この例では，正解のケースは関係なく，localized_string_type の言語は英語である
likert	IEEE ドラフトは likert 相互作用タイプに対する正解を以下の通り定義する： likert: short_identifier_type **データモデル要素実装要件:** • **データタイプ**: short_identifier_type • **値空間**: short_identifier_type を表す ISO-10646-1 文字列．詳細はセクション 4.1.1.7: *データタイプ*参照 • **フォーマット**: 文字列の詳細は： o <short_identifier_type> **LMS 動作要件:** • 各正解パターンは文字列によって表させる．文字列は上記の**データモデル要素実装要件**に準拠し，正解集合のある特定のインデックスに保存される • LMS はこのタイプの相互作用に対する正解が一つしかない場合に実施する– cmi.interactions.n.correct_responses.0.pattern．SCO が，追加のパターンを保存しようと(correct_responses index, *n*, 0 <)，SetValue()を呼び出そうとすると，LMS はエラーコートを"351" – General Set Failure に設定し "false"を返す．LMS はパターンの有効リストに新たな値を追加しない，もしくは LMS は現在保存されたパターンの状態を変更しない **例:** • SetValue("cmi.interactions.0.correct_responses.0.pattern","strongly_agree") これは正解集合が 1 つのパターンを持つ likert 相互作用の例である．API メソッドコールは，正解は"strongly_agree"であることを示す．SCO はこの値の重要性を理解する責任がある

	• SetValue("cmi.interactions.1.correct_responses.0.pattern"," likert_1") これは正解集合が 1 つのパターンを持つ likert 相互作用の例である．API メソッドコールは，正解は"likert_1"であることを示す．SCO はこの値の重要性を理解する責任がある

Matching	IEEE ドラフトは matching 相互作用タイプに対する正解を以下の通り定義する：
	
	matching:
	bag of bag of record
	{
	source:
	short_identifier_type,
	target:
	short_identifier_type,
	}
	// The bag of bag SPM: 5 bags
	// The bag of record SPM: 36 records
	
	データモデル要素実装要件:
	• **データタイプ**: 文字列
	• **値空間**: レコードのバッグを表す ISO-10646-1 文字列．各レコードは 2 個の short_identifier_types から成り立つ – 'source'および'target'．レコードの short_identifier_type は特殊なリザーブされたトークン("[.]")によって区切られている．詳細はセクション 4.1.1.7: *データタイプ*参照．バッグの各レコードは特殊なリザーブされたトークン("[.]")によって区切られている．LMS は，最低 36 レコード（要求される SPM）を含む文字列をサポートする．LMS はこれ以上をサポートすることができるが，それは実装次第である．LMS が責任があるのは SPM の 36 レコードを管理する事である
	• **フォーマット**: 文字列のフォーマットは：
	o <short_identifier_type>[.]<short_identifier_type>[,]<short_identifier_type>[.]<short_identifier_type>
	
	文字列を作るとき以下の要件が遵守されるべきである：
	
	o バッグは一つ以上のレコードを含む
	o 各レコードは特殊なリザーブされたトークン"[.]"に区切られる 'source'および'target'両方の値を持つ
	o 'source'および'target'値は short_identifier_types である
	o バッグが一つより多いレコードを含む場合，特殊なリザーブされたトークン"[,]"に区切られる
	o ある文字列に short_identifier_type が発生する回数に制限はない
	o レコードの順序は重要でない
	
	LMS 動作要件:
	• 各正解パターンは文字列によって表させる．文字列は上記の**データモデル要素実装要件**に準拠し，正解集合のある特定のインデックスに保存される
	• LMS は，正解集合に対して最低 5 (要求される SPM) パターンをサポートする．LMS はそれ以上をサポートすることが可能であるが，これは実装次第である．LMS が責任があるのは SPM の 5 を管理することだけである
	
	例:
	• SetValue("cmi.interactions.0.correct_responses.0.pattern", "1[.]a[,]2[.]c[,]3[.]b")
	これは正解集合が 1 つのパターンを持つ matching 相互作用の例である．API メソッドコールは正解は下記であることを示す：

	<table><tr><th>Source</th><th></th><th>Target</th></tr><tr><td>1</td><td>→ matches →</td><td>A</td></tr><tr><td>2</td><td>→ matches →</td><td>C</td></tr><tr><td>3</td><td>→ matches →</td><td>B</td></tr></table> SCO は正解およびその構成要素の source/target ペアの重要性を理解する責任がある

performance	IEEE ドラフトは performance 相互作用タイプに対する正解を以下の通り定義する：
	
	performance:
	bag of record
	(
	order_matters:
	boolean,
	answers:
	array(0..124) of record
	// The array of record SPM: 125 records,
	(
	step_name :
	short_identifier_type,
	step_answer :
	choice
	(
	state(literal, numeric)
	)
	of
	(
	literal :
	characterstring (iso-10646-1),
	// SPM 250
	numeric :
	record
	(
	min :
	real (10,7),
	max :
	real (10,7),
	)
	)
	)
	) // The bag of record SPM: 5 records
	
	Data Model Element Implementation Requirements:
	• **データタイプ**: 文字列
	• **値空間**: レコードを表す ISO-10646-1 文字列は {order_matters=<*boolean*>}デリミターおよびレコードのアレイから成り立つ．アレイの各レコードは以下の値の一つもしくは二つから成り立ち特殊なリザーブされたトークン(“[.]”)に区切られる：
	o ‘step name’ – short_identifier_type. 詳細はセクション 4.1.1.7: *データタイプ*参照
	o ‘step answer’ – 文字列か数値域． ‘step answer’が文字列なら LMS は最低 250 (要求される SPM)キャラクターをサポートする．LMS はそれ以上をサポートすることが可能であるが，これは実装次第である．LMS が責任があるのは SPM の 250 キャラクターを管理することだけである．‘step answer’が数地域であれば，フォーマットは：

- <min>[:]<max>

アレイの各ﾚｺｰﾄﾞは特殊なリザーブされたトークン (“[,]”)に区切られる．LMS は最低 125 (要求される SPM)レコードを含む特殊なリザーブされたトークン“[,]”でくぎられた文字列をサポートする．LMS はそれ以上をサポートすることが可能であるが，これは実装次第である．LMS が責任があるのは SPM の 125 レコードを管理することだけである．

- **フォーマット**: 文字列のフォーマットは：
  - {order_matters}<step_name>[.]<step_answer>[,]<step_name>[.]<step_answer>

文字列を作るとき以下の要件が遵守されるべきである：

- アレイは一つ以上のレコードを含む
- {order_matters=<boolean>}デリミターはアレイレコードの順序が正解パターンの評価に関係するかどうかを示す．このデリミターはオプションであり，特定されない場合， order_matters のデフォルト値は “true”である．詳細はセクション 4.1.1.6: *予約されたデリミタ*参照
- アレイの各レコードﾞは‘step_name’もしくは‘step answer’もしくは両方を含み特殊なリザーブされたトークン“[.]”に区切られる
- ‘step name’値は short_identifier_type である
- ‘step name’値はオプションである．アレイレコードが‘step name’を含まないと，アレイレコードは“[.]”が前に来る‘step answer’を持たなければ成らない
- ‘step answer’値が数値域でないなら SPM が 250 の文字列として表される
- ‘step answer’値が数値域なら，以下のフォーマットを提示する．<min>および<max>は共に実数(10, 7)データタイプである．詳細はセクション 4.1.1.7: *データタイプ*参照.
  - <min>[:]<max> –正解は供給される最小値以上で供給される最大値以下であることを示す．<min>および<max>が同一の数字であれば，正解は正にその数字である
  - [:]<max> – 正解には上限だけで下限がないことを示す．値は供給される最大値以下でなければならない
  - <min>[:] – 正解は下限だけで上限がないことをしめす．値は供給される最小値以上でなければならない
  - [:] – 上限も下限もないことを示す
- ‘step answer’値はオプションである．アレイレコードが‘step answer’を含まないと，アレイレコードは“[.]”が後に来る‘step name’を含まなければ成らない
- アレイが 1 レコード以上含めば，これらは特殊なリザーブされたトークン“[,]”によって区切られなければならない
- ある文字列に何回‘step name’が発生するかに制限はない

LMS 動作要件:

- 各正解パターンは文字列によって表させる．文字列は上記の**データモデル要素実装要件**に準拠し，正解集合のある特定のインデックスに保存される
- LMS は，正解集合に対して最低 5 (要求される SPM) パターンをサポートする．LMS はそれ以上をサポートすることが可能であるが，これは実装次第である．LMS が責任があるのは SPM の 5 を管理することだけである

例:

- SetValue(“cmi.interactions.0.correct_responses.0.pattern”, “{order_matters=false}[.]drink coffee[,][.]eat cereal”)

	これは正解集合が 1 つのパターンである performance 相互作用の例である．API メソッドコールは正解は“drink coffee” および“eat cereal”でこの順序であることを示す．API メソッドコールは‘step_name’は関係なく提供されていないことを示す • SetValue(“cmi.interactions.0.correct_responses.0.pattern”,”step_1[.]inspect wound[,]step_2[.]clean wound[,]step_3[.]apply bandage”) これは正解集合が 1 つのパターンを持つ performance 相互作用の例である．API メソッドコールは正解は“step_1”，“step_2”，“step_3”をこの順序に対応する答えで実施することを示す

sequencing	IEEE ドラフトは sequencing 相互作用タイプに対する正解を以下の通り定義する：
	sequencing:
	bag of array (0..35) of short_identifier_type,
	// The bag of array SPM: 5 arrays
	// The array of short_identifier_type SPM: 36 short_identifier_types
	データモデル要素実装要件:
	• **データタイプ**: 文字列
	• **値空間**: short_identifier_type のアレイを表す ISO-10646-1 文字列．アレイの各要素は特殊なリザーブされたトークン(“[,]”)によって区切られる．LMS は最低 36 short_identifier_types を含む文字列をサポートする．LMS はそれ以上をサポートすることが可能であるが，これは実装次第である．LMS が責任があるのは，SPM の 36short_identifier_types を管理することだけである．short_identifier_types データタイプのフォーマットに対する要件に関するより詳細情報はセクション 4.1.1.7: *データタイプ*参照
	• **フォーマット**: 文字列のフォーマットは：
	o <short_identifier_type>[,]<short_identifier_type>
	文字列を作るとき以下の要件が遵守されるべきである：
	o アレイはゼロ以上の short_identifier_type を含む
	o アレイが一つ以上の short_identifier_type (相互作用は複数の正解を持つ– 全ての正解が要求される)を含めば，各 short_identifier_type は特殊なリザーブされたトークン“[,]” で区切られる
	o 各 short_identifier_type はアレイで一度以上発生することがある
	o short_identifier_type の順序は重要ではない
	LMS 動作要件:
	• 各正解パターンは文字列によって表させる．文字列は上記の***データモデル要素実装要件***に準拠し，正解集合のある特定のインデックスに保存される
	• LMS は，正解集合に対して最低 5 (要求される SPM) パターンをサポートする．LMS はそれ以上をサポートすることが可能であるが，これは実装次第である．LMS が責任があるのは SPM の 5 を管理することだけである
	• LMS は正解のユニーク性を守る必要がある．SCO が既に正解集合に存在するアレイを保存するのに SetValue()を呼び出そうとすると，LMS はエラーコードを“351” – General Set Failure に設定し“false”を返す． LMS はパターンの有効リストに新たな値を追加しない，もしくは LMS は現在保存されたパターンの状態を変更しない
	例:
	• SetValue(“cmi.interactions.0.correct_responses.0.pattern”,“a[,]b[,]c”)
	• SetValue(“cmi.interactions.0.correct_responses.1.pattern”,“b[,]c[,]a”)
	これは正解集合が 2 つのパターンを持つ sequencing 相互作用の例である．API メソッドコールは正解は a then b then c もしくは b then c then a の順序であることを示す．

	• SetValue("cmi.interaction.0.correct_responses.1.pattern","buildHouse/buyMaterials[,]buildHouse/buildFoundation[,]buildHouse/buildFirstFloor[,]buildHouse/buildSecondFloor") これは正解集合が 2 つのパターンを持つ sequencing 相互作用の例である．API メソッドコールは正解は a then b then a の順序であることを示す.
Numeric	IEEE ドラフトは numeric 相互作用タイプに対する正解を以下の通り定義する： numeric: record (min: real (10,7), max: real (10,7)) Numeric 正解はオプションで小数点を付けられる 2 つの数字で表される．これらの数字は正解の範囲を示すのに利用されることがある．2 つの数字が同一なら，それが正解と特定される[1] **データモデル要素実装要件:** • **データタイプ**: 文字列 • **値空間**: 数値域を表す ISO-10646-1 文字列 • **フォーマット**: 文字列のフォーマットは： o <min>[:]<max> 文字列を作るとき以下の要件が遵守されるべきである： o 文字列値が数値域なら，以下のフォーマットを提示する．<min>および<max>は共に実数(10, 7)データタイプである．詳細はセクション 4.1.1.7: *データタイプ*参照. ▪ <min>[:]<max> –正解は供給される最小値以上で供給される最大値以下であることを示す．<min>および<max>が同一の数字であれば，正解は正にその数字である ▪ [:]<max> – 正解には上限だけで下限がないことを示す．値は供給される最大値以下でなければならない ▪ <min>[:] – 正解は下限だけで上限がないことをしめす．値は供給される最小値以上でなければならない ▪ [:] – 上限も下限もないことを示す **LMS 動作要件:** • 各正解パターンは文字列によって表させる．文字列は上記の**データモデル要素実装要件**に準拠し，正解集合のある特定のインデックスに保存される • LMS はこのタイプの相互作用の正解は一つしかないことを守る必要がある cmi.interactions.n.correct_responses.0.pattern.. SCO が既に正解集合に存在するパターンを保存するのに SetValue()を呼び出そうとすると，LMS はエラーコードを"351" – General Set Failure に設定し"false"を返す．LMS はパターンの有効リストに新たな値を追加しない，もしくは LMS は現在保存されたパターンの状態を変更しない **例:** • SetValue("cmi.interactions.0.correct_responses.0.pattern","4[:]10") これは正解集合が 1 つのパターンの numeric 相互作用の例である．API メソッドコールは正解は"4" から "10"の実数であることを示す • SetValue("cmi.interactions.0.correct_responses.0.pattern", "[:]10") これは正解集合が 1 つのパターンの numeric 相互作用の例である．API メソッドコールは正解は "10"以下の実数であることを示す

	• SetValue(“cmi.interactions.0.correct_responses.0.pattern”, “4[:]”) これは正解集合が 1 つのパターンの numeric 相互作用の例である．API メソッドコールは正解は“4” 以上の実数であることを示す • SetValue(“cmi.interactions.0.correct_responses.0.pattern”, “3.14159[:]3.14159”) これは正解集合が 1 つのパターンの numeric 相互作用の例である．API メソッドコールは正解は実数“10.5”であることを示す
other	IEEE ドラフトは other 相互作用タイプに対する正解を以下の通り定義する： other: characterstring (iso-10646-1) // The characterstring SPM 4000 characters **データモデル要素実装要件:** • **データタイプ**: 文字列 • **値空間**: ISO-10646-1 文字列. • **フォーマット**: other correct response は IEEE 規格で定義されない他のタイプをサポートするのに利用される．文字列のフォーマットは相互作用の実装次第である **LMS 動作要件:** • 各正解パターンは文字列によって表させる．文字列は上記の**データモデル要素実装要件**に準拠し，正解集合のある特定のインデックスに保存される • LMS はこのタイプの相互作用の正解は一つしかないことを守る必要がある cmi.interactions.n.correct_responses.0.pattern.. SCO が既に正解集合に存在するパターンを保存するのに SetValue()を呼び出そうとすると，LMS はエラーコードを“351” – General Set Failure に設定し“false”を返す．LMS はパターンの有効リストに新たな値を追加しない，もしくは LMS は現在保存されたパターンの状態を変更しない

4.2.9.2 Learner Response データモデル要素詳細

cmi.interactions.n.learner_response データモデル要素は，相互作用タイプ(cmi.interactions.n.type)によって異なる値を持つ．このセクションは，learner_response によって定義される文字列に対するデータモデル値の要件を定義する．

表 4.2.9.2a: Learner Response フォーマット要件

相互作用データタイプ	学習者レスポンスフォーマット
true_false	IEEE ドラフトは true_false 相互作用タイプに対する正解を以下の通り定義する： true_false: 状態(true, false) **データモデル要素実装要件:** • **データタイプ**: 文字列 • **値空間**: ISO-10646-1 • **フォーマット**: IEEE ドラフトは二つの状態値を定義する．SCORM はこれらの値を以下の制限された語彙トークンにバインドする： o “true”: 学習者レスポンスは true もしくは true の同義語(agree,yes など)である事を示す o “false”: 学習者レスポンスは false もしくは false の同義語

	(disagree, no など)であることを示す **例:** • SetValue(“cmi.interactions.0.learner_response”,“true”) これは学習者レスポンスが“true”の true_false 相互作用の例である
multiple_choice	IEEE ドラフトは multiple_choice 相互作用タイプに対する正解を以下の通り定義する： multiple_choice: set of short_identifier_type // The set of short_identifier_type SPM: 36 short_identifier_types **データモデル要素実装要件:** • **データタイプ**: 文字列 • **値空間**: a set of short_identifier_type を表す ISO-10646-1 文字列 （セットの各要素はリザーブされたトークン(“[,]”)によって区切られる）． LMS は最低 36 short_identifier_types を含む文字列をサポートする．LMS はそれ以上をサポートすることが可能であるが，これは実装次第である．LMS が責任があるのは，SPM の 36 short_identifier_types を管理することだけである．short_identifier_type データタイプのフォーマットに対する要件に関するより詳細情報はセクション 4.1.1.7: *データタイプ*参照 • **フォーマット**: 文字列のフォーマットは： o <short_identifier_type>[,]<short_identifier_type> 文字列を作るとき以下の要件が遵守されるべきである： o セットはゼロ以上の short_identifier_type を含む o セットが２つ以上の short_identifier_type を含めば，各 short_identifier_type は特殊なリザーブされたトークン“[,]” で区切られる o 各 short_identifier_type はセットに一度だけ発生する o short_identifier_type の順序は重要ではない **Example:** • SetValue(“cmi.interactions.0.learner_response”, “choice1[,]choice2[,]choice3”) これは，学習者レスポンスが“choice1”, “choice2”および“choice3”の choice 相互作用の例である

fill_in	IEEE ドラフトは fill_in 相互作用タイプに対する正解を以下の通り定義する：
	
	fill_in:
	{
	array (0..9) of localized_string_type(250),
	// The array SPM: 10 localized_string_type
	// The localized_string type SPM: 250 characters
	}
	
	データモデル要素実装要件:
	• **データタイプ**: 文字列
	• **値空間**: アレイの localized_string_type を表す ISO-10646-1 文字列．アレイの各要素はリザーブされたトークン(“[,]”)によって区切られる． LMS は最低 10 localized_string_types を含む文字列をサポートする．LMS はそれ以上をサポートすることが可能であるが，これは実装次第である．LMS が責任があるのは，SPM の 10 localized_string_types を管理することだけである．localized_string_type データタイプのフォーマットに対する要件に関す

	るより詳細情報はセクション 4.1.1.7: *データタイプ*参照 • **フォーマット:** 文字列のフォーマットは以下の通り： o <localized_string_type>[,]<localized_string_type> 文字列を作るとき以下の要件が遵守されるべきである： o アレイは１つ以上の localized_string_types を含む o アレイが２つ以上の localized_string_type (相互作用は複数の正解を持つ– 全ての正解が要求される)を含めば，特殊なリザーブされたトークン“[,]” で区切られる o 各 localized_string は一度以上発生する事がある o localized_strings の順序は重要でない **例:** • SetValue(“cmi.interactions.0.learner_response”, “{lang=en}car”) これは，学習者レスポンスが“car”で localized_string_type の言語が英語の fill_in 相互作用の例である • SetValue(“cmi.interactions.0.learner_response”,”car”) これは，学習者レスポンスが“car”の fill_in 相互作用の例である • SetValue(“cmi.interactions.0.learner_response”,”car[,]automobile”) これは，学習者レスポンスが“car”続いて“automobile”で localized_string_type の言語が両方とも英語の fill_in 相互作用の例である

long_fill_in	IEEE ドラフトは long_fill_in 相互作用タイプに対する正解を以下の通り定義する：
	
	long_fill_in:
	{
	localized_string_type(4000),
	// The localized_string_type SPM: 4000 characters
	}
	
	データモデル要素実装要件:
	• **データタイプ:** 文字列
	• **値空間:** localized_string_type を表す ISO-10646-1 文字列.
	localized_string_type データタイプのフォーマットに対する要件に関するより詳細情報はセクション 4.1.1.7: *データタイプ*参照
	• **フォーマット:** 文字列のフォーマットは：
	o <localized_string_type>
	
	文字列を作るとき以下の要件が遵守されるべきである：
	o 文字列は一つの localized_string_type を含まなければ成らない
	
	例:
	下記の例は Gettysburg Address 全体を含まない．スペースをセーブすること，データタイプを記述すること，だけが書かれている．実践的な利用は Gettysburg Address の全テキストを記述することである
	• SetValue(“cmi.interactions.0.learner_response”, “{lang=en}Four score and seven years ago…”)
	これは，学習者レスポンスが“Four score and seven years ago…”で localized_string_type の言語が英語の long_fill_in 相互作用の例である
	
	• SetValue(“cmi.interactions.0.learner_response”,”Four score and seven years ago…”)
	これは，学習者レスポンスが“Four score and seven years ago…”で

	localized_string_type の言語が英語の long_fill_in 相互作用の例である
likert	IEEE ドラフトは likert 相互作用タイプに対する正解を以下の通り定義する： likert: short_identifier_type **データモデル要素実装要件:** • **データタイプ**: short_identifier_type • **値空間**: short_identifier_type を表す ISO-10646-1 文字列．詳細はセクション 4.1.1.7: *データタイプ*参照 • **フォーマット**: 文字列のフォーマットは： o <short_identifier_type> **例**: • SetValue(“cmi.interactions.0.learner_response”, “strongly_disagree”) これは，学習者レスポンスが“strongly_disagree”の likert 相互作用の例である • SetValue(“cmi.interactions.1.learner_response”,”likert_1”) これは，学習者レスポンスが“likert_1”の likert 相互作用の例である

matching	IEEE ドラフトは matching 相互作用タイプに対する正解を以下の通り定義する：
	bag of record
	{
	source:
	short_identifier_type,
	target:
	short_identifier_type,
	} // The bag of record SPM: 36 records
	
	データモデル要素実装要件:
	• **データタイプ**: 文字列
	• **値空間**: レコードのバッグを表す ISO-10646-1 文字列．各レコードは‘source’および‘target’という 2 個の short_identifier_types から成り立つ．レコードの short_identifier_type は特殊なリザーブされたトークン(“[.]”)によって区切られている．詳細はセクション 4.1.1.7: *データタイプ*参照．バッグの各レコードは特殊なリザーブされたトークン(“[.]”)によって区切られている．LMS は，最低 36 レコード（要求される SPM）を含む文字列をサポートする．LMS はこれ以上をサポートすることができるが，それは実装次第である．LMS が責任があるのは SPM の 36 レコードを管理する事である
	• **フォーマット**: 文字列のフォーマットは：
	o <short_identifier_type>[.]<short_identifier_type>[,]<short_identifier_type>[.]<short_identifier_type>
	
	文字列を作るとき以下の要件が遵守されるべきである：
	
	o バッグは一つ以上のレコードを含む
	o 各レコードは特殊なリザーブされたトークン“[.]”に区切られる‘source’および‘target’両方の値を持つ
	o ‘source’および‘target’値は short_identifier_types である
	o バッグが一つより多いレコードを含む場合，特殊なリザーブされたトークン“[,]”に区切られる
	o ある文字列に short_identifier_type が発生する回数に制限はない
	o レコードの順序は重要でない
	
	例:

	• SetValue("cmi.interactions.0.learner_response", "2[.]c[,]1[.]a[,]3[.]b") **これは，学習者レスポンスが以下の通りである** matching **相互作用の例である：** **Source** | | **Target** 1 | → matches → | A 2 | → matches → | C 3 | → matches → | B SCO **は正解およびその構成要素である** source/target **ペアの重要性を理解する責任がる**

performance	IEEE **ドラフトは** performance **相互作用タイプに対する正解を以下の通り定義する：**
	performance:
	array(0..249) of record
	// The array of record SPM: 250 records
	(
	step_name:
	short_identifier_type,
	step_answer:
	choice
	(
	state(literal, numeric)
	)
	of
	(
	literal:
	characterstring (iso-10646-1)
	// The characterstring SPM: 250 characters
	numeric:
	real(10,7)
	)
	)
	
	データモデル要素実装要件:
	• **データタイプ**: 文字列
	• **値空間**: **アレイのレコードを表す** ISO-10646-1 **文字列．アレイの各レコードは以下の値の一つもしくは二つから成り立ち特殊なリザーブされたトークン**("[.]")**に区切られる：**
	o 'step name' – short_identifier_type. **詳細はセクション** 4.1.1.7: ***データタイプ*参照**
	o 'step answer' – **文字列か数値域．** 'step answer'**が文字列なら** LMS **は最低** 250 **(要求される** SPM)**キャラクターをサポートする．**LMS **はそれ以上をサポートすることが可能であるが，これは実装次第である．**LMS **が責任があるのは** SPM **の** 250 **キャラクターを管理することだけである．**'step answer'**が数地域であれば，フォーマットは実数**(10, 7)**である．実数** (10,7)**データタイプに関する詳細は，セクション** 4.1.1.7: ***データタイプ*参照．アレイの各レコードはは特殊なリザーブされたトークン** ("[,]")**に区切られる．** LMS **は最低** 250 **(要求される** SPM)**レコードを含む特殊なリザーブされたトークン**"[,]"**でくぎられた文字列をサポートする．**LMS **はそれ以上をサポートすることが可能であるが，これは実装次第である．**LMS **が責任があるのは** SPM **の** 250 **レコードを管理することだけである．**
	• **フォーマット**: **文字列のフォーマットは：**
	o <step_name>[.]<step_answer>[,]<step_name>[.]<step_answer>

	文字列を作るとき以下の要件が遵守されるべきである： o アレイは一つ以上のレコードを含む o アレイの各レコードは‘step_name’もしくは‘step answer’もしくは両方を含み特殊なリザーブされたトークン“[.]”に区切られる o ‘step name’値は short_identifier_type である o ‘step name’値はオプションである．アレイレコードが‘step name’を含まない場合，アレイレコードは“[.]”に続く‘step answer’を持たなければ成らない o ‘step answer’値が数値域でないなら SPM が 250 の文字列として表される o ‘step answer’値が数値域なら，以下のフォーマットを提示する．<min>および<max>は共に実数(10, 7)データタイプである．詳細はセクション 4.1.1.7: *データタイプ*参照. o ‘step answer’値はオプションである．アレイレコードが‘step answer’を含まない場合，アレイレコードは“[.]”が後に来る‘step name’を含まなければ成らない o アレイが２レコード以上含む場合，これらは特殊なリザーブされたトークン“[,]”によって区切られなければならない o ある文字列に何回‘step name’が発生するかに制限はない o レコードの順序は重要ではない **例:** • SetValue(“cmi.interactions.0.learner_response”,”step_1[.]inspect wound[,]step_2[.]clean wound[,]step_3[.]apply bandage”) これは，学習者レスポンスが学習者は以下のセテップを以下の順序で実施したことを示す performance 相互作用の例である： 1. step_1 → inspect wound 2. step_2 → clean wound 3. step_3 → apply bandage • SetValue(“cmi.interactions.0.learner_response”,” step_3[.]apply bandage[,]step_1[.]inspect wound[,]step_2[.]clean wound”) これは，学習者レスポンスにより学習者が以下のステップを以下の順序で実施したことが示された performance 相互作用の例である： 1. step_3 → apply bandage 2. step_1 → inspect wound 3. step_2 → clean wound
sequencing	IEEE ドラフトは sequencing 相互作用タイプに対する正解を以下の通り定義する： sequencing: array (0..35) of short_identifier_type, // The array of short_identifier_type SPM: 36 short_identifier_types **データモデル要素実装要件:** • **データタイプ**: 文字列 • **値空間:** short_identifier_type のアレイを表す ISO-10646-1 文字列．アレイの各要素は特殊なリザーブされたトークン(“[,]”)によって区切られる．LMS は最低 36 short_identifier_types を含む文字列をサポートする．LMS はそれ以上をサポートすることが可能であるが，これは実装次第である．LMS が責任があるのは，SPM の 36short_identifier_types を管理することだけである．short_identifier_types データタイプのフォーマットに対する要件に関するより詳細情報はセクション 4.1.1.7: *データタイプ*参照 • **フォーマット:** 文字列のフォーマットは： o <short_identifier_type>[,]<short_identifier_type>

	文字列を作るとき以下の要件が遵守されるべきである： o アレイはゼロ以上の short_identifier_type を含む o アレイが一つ以上の short_identifier_type (相互作用は複数の正解を持つ– 全ての正解が要求される)を含めば，各 short_identifier_type は特殊なリザーブされたトークン“[,]” で区切られる o 各 short_identifier_type はアレイで一度以上発生することがある o short_identifier_type の順序は重要ではない **Example:** • SetValue(“cmi.interactions.0.learner_response”, “a[,]b[,]c”) これは，学習者レスポンスが a then b then c の順序である sequencing 相互作用の例である • SetValue(“cmi.interaction.0.learner_response”,”b[,]c[,]a”) これは，学習者レスポンスが b then c then a の順序である sequencing 相互作用の例である
numeric	IEEE ドラフトは numeric 相互作用タイプに対する正解を以下の通り定義する： numeric: real (10,7) **データモデル要素実装要件:** • **データタイプ**: 文字列 • **値空間**: ISO-10646-1 文字列 • **フォーマット**: 文字列は実数 (10,7)を表す．実数 (10,7)データタイプに関する詳細はセクション 4.1.1.7: *データタイプ*参照 **例:** • SetValue(“cmi.interactions.0.learner_response”,“4”) これは 学習者レスポンスが“4”の numeric 相互作用の例である • SetValue(“cmi.interactions.0.learner_response”, “10.5”) これは 学習者レスポンスが“10.5”の numeric 相互作用の例である
Other	IEEE ドラフトは other 相互作用タイプに対する正解を以下の通り定義する： other: characterstring (iso-10646-1) // The characterstring SPM: 4000 characters **データモデル要素実装要件:** • **データタイプ**: 文字列 • **値空間**: ISO-10646-1 文字列 • **フォーマット**: other 学習者レスポンスは，IEEE 規格で定義されていない他のタイプの相互作用をサポートするために利用される．文字列のフォーマットは相互作用の実装次第である

4.2.10. Launch Data

学習経験中，学習アクティビティに関連付けられた SCO に，何らかの起動情報を提供する必要があることがある．これは，起動に先立って，パラメータを利用して SCO に提示することができない情報である．cmi.launch_data データモデル要素は，コンテンツ設計者にこの情報を供給する手段を提供する．

cmi.launch_data データモデル要素は，その SCO が初期化に利用できる SCO 特定のデータを提供する．このデータモデル要素の値は特定ではない [1]．

このデータモデル要素の値は，SCO の実装者によって定義される．一般的に，SCO に対するドキュメントにどんなデータが提供されうるのか，されなければならないのかが明記される [1]．

これは，典型的に起動パレメータとして SCO へ受け渡されない情報である．これは，ある特定のコンテクストにて各々の SCO の利用に必要とされる情報である．

表 4.2.10a: Launch Data データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**詳細**
cmi.launch_data	**データモデル要素実装要件:** • **データタイプ:** 文字列 (SPM: 4000) • **値空間:** ISO-10646-1 • **フォーマット:** この文字列のフォーマットは SCO 開発者に委ねられている．LMS は，SCO に要求されたら(GetValue())，単にデータを返すだけである．詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • 起動データが SCO を動かすように要求されたら，設計者はこの起動データを SCORM コンテンツアグリゲーションモデルで定義されたメカニズムを利用して提供する．ADL Content Package extension namespace は，この情報を保存する責任のある要素(<adlcp:dataFromLMS>)を提供する．この要素は，SCO リソースが参照される<imscp:item>と関連させることができる．LMS は，<adlcp:dataFromLMS> 要素で供給される値を利用して，*cmi.launch_data* 値を初期化する．要素もしくは値がなければ，LMS はこの値をどのように初期化するか仮定をしない．LMS が GetValue()要求を受け取り，マニフェストに情報が定義されていなければ，LMS は適切なエラーコードを設定する (API 実装要件参照) **SCO 動作要件:** • LMS はこのデータモデル要素を read-only として実装する必要がある．SCO は *cmi.launch_data* データモデル要素の値を検索することだけが許されている **API 実装要件:** • **GetValue():** LMS は *cmi.launch_data* データモデル要素に対して保存された値を返し，エラーコードを“0” - No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o SCO リソースが参照される<imscp:item>に対して定義された<adlcp:dataFromLMS>がなければ，LMS は *cmi.launch_data* の初期値を仮定することに責任がない．これらのケースで SCO が *cmi.launch_data* を要求すると，LMS はエラーコードを“403” - Data Model Element Value Not Initialized に設定し空の文

	字列 (“”)を返す • **SetValue():** SCO が *cmi.launch_data* データモデル要素へ SetValue()を呼び出そうとすると，LMS はエラーコードを“404”” - Data Model Element Is Read Only に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変える事はない **追加動作要件:** • SCO 開発者は *cmi.launch_data* を定義する責任がある．コンテンツパッケージマニフェストは SCO 開発者がこのデータを宣言するために利用する要素(<adlcp: dataFromLMS>)を含む．この要素は，SCO リソースを参照する<imscp:item>のサブ要素としてマニフェストにおかれている．LMS は，参照された SCO に対してこの要素で定義された値で *cmi.launch_data* を初期化する責任がある．*cmi.launch_data* はいづれの有効な表示文字列も許されている（上記のデータモデル要素実装要件で定義されたように）．LMS は最低 SPM の 4000 キャラクターを提供する．4000 キャラクター以上の文字列は LMS に全てが保存される保証はない．SCO 開発者にマニフェストでデータが定義されず，SCO が *cmi.launch_data* を要求すると，LMS は空の文字列(“”)を返し適切な API 実装エラーコードを設定する **例:** • GetValue(“cmi.launch_data”)

4.2.11. Learner Id

cmi.learner_id データモデル要素は，SCO を起動した学習者を識別する．cmi.learner_id は少なくとも SCO の範囲内では固有である[1]．どのように cmi.learner_id が割り振られるかは SCORM の対象範囲外である．どのように learner_id が割り振られるかに関する典型的な一つケースは，LMS に定義される学習者登録プロセスである．cmi.learner_id は LMS 内でで学習者を識別する．

表 4.2.11a: Learner ID データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**詳細**
cmi.learner_id	**データモデル要素実装要件:** • **データタイプ:** long_identifier_type • **値空間:** RFC 2396 [6]に従う有効な URI (Universal Resource Identifier)を表す文字列(SPM: 4000)．RFC 2141 [3]に従って URI は URN であることが推奨される • **フォーマット:** 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS は *cmi.learner_id* を初期化する責任がある．これがどのようになされるかは SCORM の対象範囲外である(これは一般的に LMS 内の学習者登録システムを通して処置される) **SCO 動作要件:** • LMS はこのデータモデル要素を read-only として実装する必要がある．SCO は *cmi.learner_id* データモデル要素の値を検索することだけが許されている **API 実装要件:** • **GetValue():** LMS は学習者に対して LMS に保存された学習者 ID を返し，エラーコードを“0” - No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する • **SetValue():** SCO が *cmi.learner_id* を設定するのに SetValue()を呼び出そうとすると，LMS はエラーコードを“404”” - Data Model Element Is Read Only に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変える事はない **例:** • GetValue(“cmi.learner_id”)

4.2.12. Learner Name

cmi.learner_name データモデル要素は，LMS によって学習者に提供される名前である[1]． どのように cmi.learner_name が割り振られるかもしくは作成されるかは SCORM の対象範囲外である．cmi.learner_name は，LMS 学習者登録システムから来る，もしくは学習者プロフィール情報から来ることがある．cmi.learner_name の作成および割付に対して他のメカニズムがあることがあるが，どのようにこのプロセスが達成されるかに関して制約はない．

LMS もしくは SCO が，文字列に対するデリミターをどのように解釈するかもしくは処理するかは，SCORM の対象範囲外である．SCO は，LMS に返された文字列を利用する前に言語情報を分析するように構築されることもある．

表 4.2.12a: Learner Name データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.learner_name	**データモデル要素実装要件:** • **データタイプ:** localized_string_type (SPM: 250) • **値空間:** ローカライズされた文字列を表す文字列 • **フォーマット:** 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS は *cmi.learner_name* を初期化する責任がある．これがどのようになされるかは SCORM の対象範囲外である(これは一般的に様々なメカニズムで処置される) **SCO 動作要件:** • LMS はこのデータモデル要素を read-only として実装する必要がある．SCO は *cmi.learner_name* データモデル要素の値を検索することだけが許されている • GetValue()要求中，SCO はデリミターは LMS から返された文字列の 1 部であるかもしれないことに留意するべきである（LMS の実装による）．SCO が LMS に返された文字列に何をするかは SCO の実装次第である．LMS にデリミターが提供されないなら，SCO は文字列はデフォルト値(“en”)を利用していると仮定する **API 実装要件:** • **GetValue():** LMS は学習者に対して LMS に保存された関連する *cmi.learner_name* を返し，エラーコードを“0” - No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する • **SetValue():** SCO が *cmi.learner_name* を設定するのに SetValue()を呼び出すと，LMS はエラーコードを“404”” - Data Model Element Is Read Only に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変える事はない **例:** • GetValue(“cmi.learner_name”)

4.2.13. Learner Preference

学習者のプリファレンスデータは，学習者の SCO の利用に関連付けられた学習者が指定するプリファレンスである [1]．この学習者のプリファレンスデータがどのように決定されるかについては制約はない．例えば，LMS は，学習者がこのデータを設定できる何らかのインターフェースを提供することができる，もしくは，学習者に関する情報を捉える何らかのプロフィールシステムを提供することができる．このようなメカニズムができたら，このグローバルに定義されたデータは，このセクションで定義されるデータモデル要素を初期化するのに利用されるべきである．このメカニズムがどのように実装されるかによって，データはあるコンテンツ組織で全ての SCO に適用されることもあり，SCO のサブセットもしくは SCO 単体に適用されることもある．

SCO のランタイム実行の前に，学習者のプリファレンスの情報を決定し設定するメカニズムがあれば，ある特定の要件が遵守される必要がある．このデータは，影響を受ける SCO にグローバルとみなされるので（コンテンツ組織内の全ての SCO に適用する場合があるなど），学習者の試行中に SetValue()コールを経由して，SCO が学習者のプリファレンスを設定しても，データは変更されるべきでない．学習者のプリファレンスが，学習者の試行中に設定されても，このデータは，SCO へのその試行に対してだけ利用可能である（学習者の試行に対して，一時的にグローバルに定義されたプリファレンスを無効にするなど）．新規の試行が始まると，SCO に設定されたいかなる学習者のプリファレンスも，前回の学習者の試行から失われる．値は，下記に定義されるようにデフォルト値に初期化されるかもしくは他の方法で得られたグローバル値に戻される．

SCO のランタイム実行の前に，学習者のプリファレンスの情報を決定するメカニズムがなければ，デフォルト値もしくはデフォルト動作が，各データモデル要素に対して定義されている．SCO に設定されたデータの保持に関しても同じ要件が適用する．そのデータはその SCO への学習者の試行に対してのみ存在する．

表 4.2.13a: Learner Preference データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.learner_preference._children	*cmi.learner_preference._children* データモデル要素は，サポートされるデータモデル要素のリストを表す．このデータモデル要素は，一般的に LMS にサポートされるデータモデル要素を特定するのに SCO に利用される．返された文字列は GetValue()および SetValue()要求に対するパラメータを動的に作成することに利用される **データモデル要素実装要件:** • **データタイプ:** characterstring • **値空間:** ISO-10646-1 [5] • **フォーマット:** 基の学習者プリファレンスデータモデル要素内の LMS にてサポートされている全てのデータモデル要素をコンマで区切ったリスト．全てのデータモデル要素は LMS にサポートされるように要求されているので，文字列は以下のデータモデル要素を表す： o audio_level o language o delivery_speed o audio_captioning **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS はコンマで区切られた全てのデータモデル要素のリ

	ストを返す責任がある (上記データモデル要素実装要件参照) **SCO 動作要件:** • LMS はこのデータモデル要素を read-only.として実装する必要がある．SCO は *cmi.learner_preference._children* データモデル要素の値を検索することだけが許されている **API 実装要件:** • **GetValue():** LMS はコンマで区切られた LMS にサポートされるデータモデル要素のリストを返し(上記データモデル要素実装要件参照)，エラーコードを“0” – No error に設定する．データモデル要素の順序は重要でない．返された文字列はデータモデル要素実装要件で特定された要件を遵守する • **SetValue():** *cmi.learner_preference._children* を設定するために，SCO が SetValue()要求を呼び出すと，LMS はエラーコードを“404” – Data Model Element Is Read Only に設定し “false”を返す **例:** • GetValue(“cmi.learner_preference._children”)
cmi.learner_preference.audio_level	*cmi.learner_preference.audio_level* データモデル要素は知覚されるオーディオレベルの意図的な変更を指定する multiplier 値で，知覚されるオーディオレベルは，レベル 1 が「変化なし」を意味する実装特化参照レベルを表す．例えば，値 0 無限大の弱化，値 0.5 は 10 デシベルの弱化，値 2 は 10 デシベルの増幅を定義する[1] **データモデル要素実装要件:** • **Data Type:** 実数(10,7), 幅 (0..*) • **値空間:** 0 以上の実数 • **フォーマット:** 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS Behavior Requirements:** • このデータモデル要素は必須で LMS に read/write として実装される • デフォルトの *cmi.learner_preference_audio_level* は 1 である **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.learner_preference.audio_level* データモデル要素の値を検索および検索することが許されている • SCO が学習者アテンプト中に *cmi.learner_preference.audio_level* を設定すると，この値はこの学習者アテンプト中しか LMS に保持されない **API 実装要件:** • **GetValue():** LMS は学習者に対して LMS に保存された関連する *cmi.learner_preference.audio_level* を返し，エラーコードを“0” - No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o SCO が，値が SCO もしくは他の方法で設定される前に，*cmi.learner_preference.audio_level* を得る要求を呼び出すと，LMS はデフォルト値の“1”

	を返しエラーコードを“0” – No error に設定する • **SetValue():** LMS は *cmi.learner_preference.audio_level* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータへ設定し，エラーコードを“0” - No error に設定し“true”を返す　LMS はこの要求に基づいてデータモデル要素の状態を変える事はない o SCO が *cmi.learner_preference.audio_level* を実数でない値に設定すると，LMS はエラーコードを“406” - Data Model Element Type Mismatch に設定し“false”を返す LMS はこの要求に基づいてデータモデル要素の状態を変える事はない o SCO が *cmi.learner_preference.audio_level* を実数に設定するが値が 0 より大きくないと(範囲外), LMS はエラーコードを“407” Data Model Element Value Out Of Range に設定する．LMS はこの要求に基づいてデータモデル要素の状態を変える事はない **Example:** • GetValue(“cmi.learner_preference.audio_level”) • SetValue(“cmi.learner_preference.audio_level”,”3”)
cmi.learner_preference.language	*cmi.learner_preference.language* データモデル要素は，複数言語機能を持つ SCO に対して学習者が選択した言語である[1] **Data Model Element Implementation Requirements:** • **データタイプ:** language_type (SPM 250) • **値空間:** iso-646 [4] • **フォーマット:** 詳細はセクション 4.1.1.7: *データタイプ*参照．デフォルト言語は “” (空の文字列)である **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • LMS が学習者のプリファレンスを初期化するメカニズムを提供しないと，*cmi.learner_preference.language* のデフォルト値は空の文字列 （“”）である **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.learner_preference.language* データモデル要素の値を検索および検索することが許されている • *cmi.learner_preference.language* のデフォルト値が空の文字列なので，この値の意味は SCO に決定される．例えば，空の文字列はデフォルト値は英語であると意味することができる．空の文字列は選択された言語はなく，SCO が学習者が選択した言語があるかどうかを決定するように作成されることを意味する． • SCO が学習者アテンプト中に *cmi.learner_preference.language* を設定すると，この値はこの学習者アテンプト中しか LMS に保持されない **API 実装要件:** • **GetValue():** LMS は学習者に対して LMS に保存された関連する *cmi.learner_preference.language* を返し，エラーコードを“0” - No error に設定する．返された文字列はデー

	タモデル要素実装要件で特定された要件を遵守する o SCO が，値が SCO もしくは他の方法で設定される前に，*cmi.learner_preference.language* を得る要求を呼び出すと，LMS はエラーコードを“0” – No error に設定し，デフォルト値の空の文字列（“”）を返す • **SetValue():** LMS は *cmi.learner_preference.language* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータへ設定し，エラーコードを“0” - No error に設定し“true”を返す o SCO が *cmi.learner_preference.language* をデータモデル要素実装要件に満たない値に設定しようとすると，LMS はエラーコードを“406” - Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変える事はない **例:** • GetValue(“cmi.learner_preference.language”) • SetValue(“cmi.learner_preference.language”,”fr-CA”)
cmi.learner_preference.delivery_speed	*cmi.learner_preference.delivery_speed* データモデル要素は，学習者が好むコンテンツ配信の相対的なスピートで，スピードの変化は実装特化参照スピードに相対的に表現される．例えば，2 は参照スピードの 2 倍の速さで 0.5 は参照スピードの半分である．デフォルト値は 1 である[1]. **データモデル要素実装要件:** • **データタイプ:** 実数(10,7), 範囲 (0..*) • **値空間:** 0 以上の実数 • **フォーマット:** 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • LMS が学習者のプリファレンスを初期化するメカニズムを提供しないと，*cmi.learner_preference.delivery_speed* のデフォルト値は “1”である **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.learner_preference.delivery_speed* データモデル要素の値を検索および検索することが許されている • SCO が学習者アテンプト中に *cmi.learner_preference.delivery_speed* を設定すると，この値はこの学習者アテンプト中しか LMS に保持されない **API 実装要件:** • **GetValue():** LMS は学習者に対して LMS に保存された関連する *cmi.learner_preference.delivery_speed* を返し，エラーコードを“0” - No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o SCO が，値が SCO もしくは他の方法で設定される前に，*cmi.learner_preference.delivery_speed* を得る要求を呼び出すと，LMS はデフォルト値の

	“1” を返しエラーコードを“0” – No error に設定する • **SetValue():** LMS は *cmi.learner_preference.delivery_speed* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータへ設定し，エラーコードを“0” - No error に設定し“true”を返す o SCO が *cmi.learner_preference.delivery_speed* を実数でない値に設定すると，LMS はエラーコードを“406” - Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変える事はない o SCO が *cmi.learner_preference.delivery_speed* を実数に設定するが値が 0 より小さいと(範囲外), LMS はエラーコードを“407” Data Model Element Value Out Of Range に設定する．LMS はこの要求に基づいてデータモデル要素の状態を変える事はない **追加動作要件:** SCO は学習者のプリファレンスに基づいてスピード値を設定する責任がある．どのようにこの値が収集され，決定され SCO に適用されるかは SCORM の対象範囲外である **例:** • GetValue(“cmi.learner_preference.delivery_speed”) • SetValue(“cmi.learner_preference.delivery_speed”,”0.5”)
cmi.learner_preference.audio_captioning	*cmi.learner_preference.audio_captioning* データモデル要素は，オーディオに対応する見出しのテキストが表示されるかどうかを示す[1] **データモデル要素実装要件:** • **データタイプ:** 状態(off, no_change, on) • **値空間:** IEEE ドラフトは三つの状態値を定義する．SCORM はこれらの状態値を以下の制限された語彙トークンにバインドする： o “-1”: “off” 状態を表す．見出しはなく，オーディオに対応するテキストは表示されない[1] o “0”: “no_change”状態を表す．現在のデフォルト見出し設定が維持される[1]．これがデフォルト値である o “1”: “on” 状態を表す．見出しが写され，オーディオに対応するテキストが表示される[1] **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.learner_preference.audio_captioning* データモデル要素の値を検索および検索することが許されている • SCO が学習者アテンプト中に *cmi.learner_preference.audio_captioning* を設定すると，この値はこの学習者アテンプト中しか LMS に保持されない **API 動作要件:** • **GetValue():** LMS は学習者に対して LMS に保存された関

	連する *cmi.learner_preference.audio_captioning* を返し，エラーコードを“0” - No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o SCO が，値が SCO もしくは他の方法で設定される前に，*cmi.learner_preference.audio_captioning* を得る要求を呼び出すと，LMS はデフォルト値の“0” を返しエラーコードを“0” – No error に設定する • **SetValue():** LMS は *cmi.learner_preference.delivery_speed* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータへ設定し，エラーコードを“0” - No error に設定し“true”を返す o SCO が *cmi.learner_preference.audio_captioning* を上記の制限された語彙トークンにない値に設定すると，LMS はエラーコードを“406” - Data Model Element Type Mismatch に設定し“false” を返す．LMS はこの要求に基づいてデータモデル要素の状態を変える事はない **追加動作要件:** このデータモデル要素がどのように初期化されるかは SCORM の対象範囲外である．値はいくつかの方法で初期化される．例えば： • SCO はが，学習者のプリファレンスに基づいて，オーディオ見出し初期値を設定するように作成されることがある，もしくは • 何らかの学習者のプリファレンスもしくは LMS に収集されたプロフィールデータに基づいて値が初期化されることがある どのようにこの値が収集され，決定され，SCO に適用されるかは SCORM の対象範囲外である **例:** • GetValue(“cmi.learner_preference.audio_captioning”) • SetValue(“cmi.learner_preference.audio_captioning”,”-1”)

4.2.14. Location

cmi.location データモデル要素は，SCO 内でのロケーションである．その値と意味は SCO によって定義される．学習者が最初に SCO に試行するとき，もしくは好ましい初期ロケーションがなければ，値は空の文字列(“”)である [1].

LMS はこのデータを解釈もしくは変更すべきでない．このデータは，LMS にとっては不明瞭である．cmi.location のフォーマットは，値を設定する SCO によってのみ理解される．SCO が cmi.location を交信すると，このデータモデル要素は，SCO 内のブックマークもしくはチェックポイントを示すのに利用される．このデータモデル要素は，中断された学習セッションの後，SCO への再エントリーに関する開始点として利用されることがある．このケースでは，cmi.location データモデル要素は，学習者が最後に SCO を経験したときの SCO 終了点と対応する．このデータモデル要素の動作と利用は SCO に管理される．

表 4.2.14a: Location データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.location	**データモデル要素実装要件:** • **データタイプ:** 文字列 (SPM: 1000) • **値空間:** ISO-10646-1 [5] • **フォーマット:** この文字列のフォーマットは SCO 開発者次第である．LMS は，SCO に要求(SetValue())されたら，単にデータを保存し，SCO に要求(GetValue())されたらデータを返す．詳細はセクション 4.1.1.7: *データタイプ*参照. **LMS Behavior Requirements:** • このデータモデル要素は必須で LMS に read/write として実装される • このデータモデル要素は SCO に管理される．初期化ステップは LMS に要求されていない．SCO に値が設定される前に GetValue()要求が呼び出されると，LMS は下記の API 実装要件に従って動作する **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.location* データモデル要素の値を検索および保存することが許されている • このデータモデル要素が保存され，値は SCO 実装で定義されると，文字列のファーマットは ISO 10646-1 に準拠する．LMS は SCO に要求されたときに単にデータを保存し返すことだけに責任がある **API 動作要件:** • **GetValue():** LMS は学習者に対して LMS に保存された関連する *cmi.location* を返し，エラーコードを“0” - No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o SCO が，値が SCO に初期化される前に，*cmi.location* を要求すると，LMS は空の文字列(“”) を返しエラーコードを “403” – Data Model Element Value Not Initialized に設定する • **SetValue():** SCO が *cmi.location* を設定するのに SetValue()要求を呼び出し，供給された値がデータモデル要素実装要件で定義された要件を満たすと，LMS は *cmi.location* に対する供給された値を保存しエラーコードを“0” – No error に設定し“true”を返す **例:** • GetValue(“cmi.location”)

	• SetValue(“cmi.location”,”chkPt1.p3.f5”)

4.2.15. Maximum Time Allowed

cmi.max_time_allowed データモデル要素は，学習者の試行の際に，学習者が SCO を利用するのに許された累積時間である[1]．学習者の試行は，最初の学習者セッションの開始から始まり、アクティビティが終了するまで続く．

表 4.2.15a: Maximum Time Allowed データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.max_time_allowed	**データモデル要素実装要件:** • **データタイプ**: 時間インターバル (秒,10,2) – 精度 0.01 秒の時間のインターバル • **フォーマット**: 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS は，SCO 開発者に提供される値に基づいて，この値を初期化することに責任がある．この値はコンテンツ組織と関連するコンテンツパッケージマニフェストで提供されることがある．LMS は，この値を初期化するのに，マニフェストで SCO リソースを参照する<imscp:item>によって定義されている場合，<imsss:attemptAbsoluteDurationLimit>要素を利用する **SCO 動作要件:** • LMS はこのデータモデル要素を read-only として実装する必要がある．SCO は *cmi.max_time_allowed* データモデル要素の値を検索することだけが許されている **API Implementation Requirements:** • **GetValue():** LMS は *cmi.max_time_allowed* データモデル要素に保存された値を返し，エラーコードを“0” - No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o マニフェストに定義された最長時間がなく，SCO がこのデータモデル要素に対して値を要求すると，LMS はエラーコードを“403” – Data Model Element Value Not Initialized に設定し空の文字列(“”) を返す • **SetValue():** *cmi.max_time_allowed* を設定するために，SCO が SetValue()要求を呼び出すと，LMS はエラーコードを “404” – Data Model Element Is Read Only に設定し “false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない **例:** • GetValue(“cmi.max_time_allowed”)

4.2.16. Mode

cmi.mode データモデル要素は，SCO が，学習者より提示される 3 つの可能なモードの一つを識別する[1]．この値は，起動後の SCO 動作を示すのに利用される．多くの SCO は単一動作を持つ．しかしながら，いくつかの SCO は，異なった量の情報を提示したり，異なった順序で情報を提示したり，異なったトレーニング戦略を反映する情報を提示したり，あるいは，SCO が現在持つモードに基づいて異なったセットのデータを保存することがある．

表 4.2.16a: Mode データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.mode	**<u>データモデル要素実装要件:</u>** • **データタイプ:** 状態(browse, normal, review) [1] • **値空間:** IEEE ドラフトは 3 つの状態値を定義する．SCORM はこれらの状態値を下記の制限された語彙トークンにバインドする： o “browse”: SCO は現在の学習者セッションに関する情報を記録する意図なしに提示される[1] o “normal”: SCO は現在の学習者セッションに関する情報を記録する意図をもって提示される[1]．これは，モードを特定するメカニズムがなければ，デフォルト値である． o “review”: SCO は，学習者アテンプトに関する情報を以前に記録しており， この情報を現在の学習セッションからのデータでアップデートする意図なしに提示される[1] **<u>LMS 動作要件:</u>** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • SCO のモードを特定するメカニズムはない．これは現在 LMS の実装に任されている．LMS がコンテンツ組織をプレビューもしくはレビューする方法を提供したければ，これは，コンテンツ組織でコンテンツ（SCO）が閲覧されるモードの *cmi.mode* を初期化するメカニズムの一つである．“normal”モードが全ての SCO に対するデフォルトモードである **<u>シーケンシングインパクト:</u>** • *cmi.mode* データモデル要素はシーケンシングに影響がない • *cmi.mode* の値はどのようなシーケンシング動作にも影響を与えない．*cmi.mode* 値が“browse” もしくは“review”なら，LMS は SCO に送られたどんなデータも（シーケンシング決定をするために）情報的として扱う．Review もしくは browse モードであるときに， SCO に送られたデータを LMS が保持するかどうかは SCORM の対象範囲外である **<u>SCO 動作要件:</u>** • LMS はこのデータモデル要素を read-only として実装する必要がある．SCO は *cmi.mode* データモデル要素の値を検索することだけが許されている **<u>API 実装要件:</u>** • **GetValue():** LMS は学習者に対して LMS に保存された関連する *cmi.mode* を返し，エラーコードを“0” - No error に設定する．異なったモードをサポートするメカニズムがなければ，LMS は全てのケースで“normal”を返す • **SetValue():** *cmi.mode* を設定するために，SCO が要求を呼び出すと，

	LMS はエラーコードを“404” – Data Model Element Is Read Only に設定し “false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない **例:** • GetValue(“cmi.mode”)

4.2.16.1 Mode および Credit 利用要件

cmi.mode および cmi.credit データモデル要素はお互いに関係がある．以下の表はその関係を表す．:

表 4.2.16.1a: Mode および Credit 値

cmi.mode 値	*cmi.credit* 値
“browse”	“no-credit”
“review”	“no-credit”
“normal”	“credit”もしくは“no-credit”

LMS は，これらのデータモデル要素を read-only として実装するように要求される．SCO は，これらの値の状態を SetValue() API コールで修正することを許されていない．LMS は，これらの値がお互いに同期が取れていることを保証しなければならない．クレジットが “no-credit” であれば，SCO によって交信されたいかなる情報も (success_status, interactions, など)有益な情報である．データはシーケンシング決定をするのに利用される．それ以外，は LMS は，自由にデータと何をしてもよい．

4.2.17. Objectives

インストラクショナルデザイナーは，学習もしくは他の種類の目標を学習アクティビティおよび関連するコンテンツオブジェクトに関連させたいと望むことがある．SCORM は，何が目標なのかを定義しない，もしくは，その利用に要件を要請しない．しかし，SCORM は，種類に関わらず，学習者が経験している間に，どのように目標のステータスがトラッキングされるかを定義し、どのようにトラッキングされた目標スレータスがシーケンシング評価に影響を及ぼすかを定義する．

学習者経験のために，識別子を通して目標ステータス情報のセットと各々のトラッキングされた目標を関連づけることで目標がトラッキングされる．利用される識別子はいかなる意味も持たない．識別子は学習者経験の結果をコンテンツ開発者に定義された目標に関連づけるのに利用されるだけである．

目標は，ある SCO に対する目標ステータス情報の集合として扱われる．目標ステータス情報は，学習もしくは SCO に関連づけられた他の種類の目標をトラッキングするために利用される．SCO は，ゼロ以上の目標ステータス情報を持つ．目標の親データモデル要素の中で追跡された情報は，SCO にだけ入手可能であり，学習者が試行している間は，入手可能である．識別子は，目標ステータス情報のセットを差別化するように要求される．ある SCO に対して，全ての目標識別子は，固有でなければならない．

目標ステータス情報の各セットは，以下の要素から成り立つ：

- 識別子：目標ステータス情報のセットに対する識別子
- スコア：スコア
- 成功ステータス：目標に対する成功ステータスを示す
- 完了ステータス：目標に対する完了ステータスを示す
- 進捗度: 関連する目標の達成に向けた度合い
- 説明：目標の簡単で有益な情報説明

学習者が SCO と相互作用している間，SCO は，ステータス情報/スコア情報を更新することを許されており，アレイに新規のエントリーを作成する必要はない．目標アレイは，学習者が相互作用している間，目標のステータスとしての役割を果たすべきである．目標識別子は，目標を固有にするものである．従って，新しい目標ステータス情報がトラッキングされる必要があるとき，アレイに新規のエントリーが作られるべきである．

SCO 開発者は，SCO のトラッキングされた目標，つまり，目標識別子の重要性を SCORM メタデータの分類カテゴリーを利用することによって表すことができる．Purpose, Taxonpath および Description 要素も，SCO の目標を表し，識別するのに利用できるが，メタデータを SCO に適用するのは，有益な情報目的に限られる．現在，LMS が，SCO に関連付けられたメタデータを処理するのに，定義された要件はなく，また，LMS に，SCO に関連付けられたメタデータを相互運用的に利用するために，定義された動作もない（詳細は SCORM CAM 参照）．

LMS は，最低 SPM の 100 目標ステータス情報をサポートする．LMS は，自由に SPM 以上をサポートする．

4.2.17.1 SCO に目標ステータス情報を関連つける

二組以上の目標ステータス情報が，一つの SCO に対して関連付けられることがある． 複数のセットの目標ステータス情報にアクセスするために定義されたメカニズムは，インデックスリストである（セクション 4.1.1.3: *集合の扱い*参照）．目標ステータス情報を設定（SetValue()）するとき，SCO は，目標ステータス情報をシーケンシャル順に挿入する役割がある．

表 4.2.17.1a: 集合データ保存に対するシナリオ

誤り	正しい
cmi.objectives.0.id = “ID1” cmi.objectives.2.id = “ID3” cmi.objectives.1.id = “ID2”	cmi.objectives.0.id = “ID1” cmi.objectives.1.id = “ID2” cmi.objectives.2.id = “ID3”

上記の，表 4.2.1.7a，の誤ったシナリオを避けるため，SCO は，インデックスリストで次に利用可能なポジションを検索するための，cmi.objectives._count に GetValue()を呼び出すことができる．LMS は，このリストをゼロベースのインデックスリストとして管理する役割がある．SCO が，cmi.objectives._count を要求するとき，LMS は，現在 LMS によって管理される目標の合計数量を返す．例えば，LMS が GetValue(“cmi.objectives._count”)要求から“2”を返すと，SCO は，この値を次のセットの目標情報を挿入するのに利用する．“2”は，SCO は既に cmi.objectives.0 および cmi.objectives.1 目標情報を設定していることを示す．

インデックスリストで，目標ステータス情報の順序は，目標間の重要な関係を定義しないし，順序は学習者セッション間で変わることがある．目標ステータス情報にアクセスするのに推奨される方法は，インデックスリストを探索して，望まれる目標識別子を探し当てることである．識別子を含むインデックスは，目標ステータス情報の他の要素にアクセスし修正するために利用されるべきである．

4.2.17.2 シーケンシングに対する目標ステータスの利用

インストラクショナルデザイナーは，条件付のシーケンシング決定をするのに，目標を利用したいと思うことがある．この要望は，学習アクティビティに関連付けられたシーケンシング情報に，明示的に表われる(SCORM CAM ブック参照)． SCO が，目標を識別子を通して参照するように，学習アクティビティに関連付けられたシーケンシング情報も同様である．SCORM は，学習経験中に，シーケンシング目的で学習アクティビティに定義された目標と SCO に利用可能な目標を関連させるのに，どのように識別子が利用されるか定義する．

SCORM は，学習経験の間にシーケンシングのために学習アクティビティで定義された目標をアクティビティの関連 SCO が利用可能な目標の状態情報(cmi.objectives)に関連付けるために識別子がどのように使われるか，を定めている．SCO が新しい学習アテンプトのために起動されると，LMS はランタイム学習目標(cmi.objectives)のセットを SCO に関連する学習アクティビティのシーケンシングのために管理される目標の状態情報によって初期化する．LMS は，SCO に関連するアクティビティに適用されるシーケンシング情報(<imsss:objectives>の子要素)の中で定義された各々の学習目標のために，SCO の学習目標コレクション中に 1 つのエントリを作成する．シーケンシング情報はランタイム学習目標を以下のように初期化するために用いられる．

1. cmi.objectives.n.id は学習目標の ID(学習目標 ID の属性はアクティビティのシーケンシング情報において学習目標に関連する)によって初期化される．
2. cmi.objectives.n.success_status は学習目標(おそらく学習目標マップの読み込みを通した)に関連するトラッキング情報で初期化される．
3. cmi.objectives.n.score.scaled は学習目標(おそらく学習目標マップの読み込みを通した)に関連するトラッキング情報で初期化される．

SCO の新しい学習アテンプトが始まり SCO の学習目標データモデル要素の初期化が LMS で行われた後，LMS に保持される学習目標の数(cmi.objectives._count)は，アクティビティがトラッキングされるか否

かに関わらず, SCO に関連する学習目標 ID を定義したアクティビティに適用されるシーケンシング情報(<imsss:objectives>要素の子)で定められたた学習目標の数と等しい.

SCO 上の以降の学習セッションが同じ学習アテンプトの中で始まる時, LMS はシーケンシングの実行において利用できる最も直近の情報を用い, SCO の学習目標データモデル要素(cmi.objectives.n.xxx)をアクティビティのシーケンシング情報で定義された学習目標のために更新する. 前回の学習セッションの最中に SCO によって作られた付加的な学習目標は影響を受けない.

ADL Note:

- SCO 上の学習アテンプトが中断されその後に再開された場合, LMS は SCO の学習目標データモデル要素の値をアクティビティの学習目標に関連する全ての SCO に中断中に起こったシーケンシングのトラッキング情報の変化を反映するような更新を行うように求められる.
- SCO の学習目標データモデル要素は SCO に関連するアクティビティがトラッキングされなくても(tracked = False)初期化される. これらの場合, 'read' 学習目標マップは学習目標の状態を初期化するために適用される. 'read' 学習目標マップが定義されない場合, 初期化された学習目標はデフォルトの(unknown)状態を持つ.
- 目標(cmi.objectives.n.xxx)に関連するランタイムデータは, 目標 ID 属性が, シーケンシング情報内で定義されない限り, アクティビティの関連する SCO に対して初期化されない. さらに, 一つのアクティビティに対して複数の目標が定義されると, SCO の目標(cmi.objectives.n.xxx)の順序は, アクティビティの目標の順序と一致するとは限らない–ID が重要なだけである.

表 4.2.17a: Objectives モデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	説明
cmi.objectives._children	*cmi.objectives._children* データモデル要素は, サポートされるデータモデル要素のリストを表す. このデータモデル要素は一般的にどのデータモデル要素が LMS にサポートされているか特定するのに SCO に利用される. 返された文字列は, GetValue()および SetValue()要求に対してパラメータを動的に作成するのに SCO に利用されることがある **データモデル要素実装要件:** • **データタイプ**: 文字列 • **値空間**: ISO-10646-1 [5] • **フォーマット**: LMS にサポートされた親対話データモデル要素の全てのデータモデル要素のコンマで区切られたリスト. 全てのデータモデル要素は LMS にサポートされるように要求されているので, 文字列は以下のデータモデル要素を表す: o id o score o success_status o completion_status o description **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS はコンマで区切られた全てのデータモデル要素のリストを返す責任がある (上記データモデル要素実装要件

	参照) **SCO 動作要件:** • LMS はこのデータモデル要素を read-only.として実装する必要がある．SCO は *cmi.objectives._children* データモデル要素の値を検索することだけが許されている **API 実装要件:** • **GetValue():** LMS はコンマで区切られた LMS にサポートされるデータモデル要素のリストを返し(上記データモデル要素実装要件参照)，エラーコードを“0” – No error に設定する．データモデル要素の順序は重要でない．返された文字列はデータモデル要素実装要件で特定された要件を遵守する • **SetValue():***cmi.objectives._children* を設定するために，SCO が SetValue()要求を呼び出すと，LMS はエラーコードを“404” – Data Model Element Is Read Only に設定し“false”を返す **例:** • GetValue(“cmi.objectives._children”)
cmi.objectives._count	*cmi.objectives._count* keyword は LMS に保存される目標の現在の数を記述する．LMS に現在保存されるエントリーの合計数が返される．LMS は 100 目標の SPM をサポートする必要がある **データモデル要素実装要件:** • **データタイプ**: 整数 • **値空間**: 負でない整数 • **フォーマット**: LMS が現在保持する目標の数を表す文字列 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS が，目標が設定される前に，*cmi.objectives._count* 値を得る要求を受け取ると，LMS は下記の API 実装要件を遵守する **SCO 動作要件:** • LMS はこのデータモデル要素を read-only.として実装する必要がある．SCO は *cmi.objectives._count* データモデル要素の値を検索することだけが許されている **API 実装要件:** • **GetValue():** LMS は LMS に現在保存される目標の数を返しエラーコードを“0” – No error に設定する o SCO に対して目標情報が利用可能になるまで，LMS は“0”を返す．これは現在保存されている目標情報がないことを意味する • **SetValue():** *cmi.objectives._count* を設定するために，SCO が SetValue()要求を呼び出すと，LMS はエラーコードを“404” – Data Model Element Is Read Only に設定し “false” を返す • **例:** • GetValue(“cmi.objectives._count”)
cmi.objectives.n.id	*cmi.objectives.n.id* データモデル要素は学習目標に対する識別子で

	あり，少なくとも SCO の範囲内でユニークである． *cmi.objectives.n.id* データモデル要素は，下記にて記述されているスコアやステータスが実装されているのであれば，有効な値を含む[1]．SCO が目標情報を保存するように要求すれば，SCO は他の目標情報の前に識別子を最初に設定するように要求される．いったん *cmi.objectives.n.id* が値を持てば，データモデル要素は違う値に再設定されることは許されない． **データモデル要素実装要件:** • **データタイプ:** long_identifier_type • **値空間:** RFC 2396 [6]に基づく有効 URI を表す文字列 (SPM: 4000)．URI は RFC 2141 [3]に基づく URN であることが推奨される • **フォーマット:** より詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • <imsss:objectives>がコンテンツパッケージマニフェストで<imscp:item>要素に対して定義されると，LMS は，学習アクティビティに参照および管理された *Objective Progress Information* に基づいて，SCO に対する目標ランタイムデータ(*cmi.objectives.n.xxx*)を初期化する責任がある．目標(cmi.objectives.n.xxx)に関連するランタイムデータは，目標 ID 属性がシーケンシング情報で定義されていない限り(<imsss:primaryObjective> or <imsss:objective>)，アクティビティに関連する SCO に対して初期化されるべきでない．目標 ID 属性は *cmi.objectives.n.id* 値を初期化するのに利用されるべきである．マニフェストで定義された目標の数は初期化される必要のある目標ステータス情報の数を決定する．LMS は，目標情報データに対するステータスとスコアが LMS に入手可能であれば，この情報を初期化する責任がある（詳細は SCORM SN ブック－グローバル目標参照） **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.objectives.n.id* データモデル要素の値を検索および保存することが許されている • SCO が目標情報を保存するように要求した場合，SCO は ID が一つの目標を他の目標からユニークに識別するように ID が設定されていることを確実にしなくてはならない．ID は最初に (他の手段で初期化されていない限り)他のどんな目標情報よりも前に設定される • SCO は，学習者アテンプト中に，既存の目標 ID を変えないことが推奨される．SCO が学習者アテンプト中に目標 ID を変更すると，前の学習者セッションで収集された目標データが損なわれることがあり，LMS になされるシーケンシング決定に影響を与えることがある **API 実装要件:** • **GetValue():** LMS は，学習者に対して現在 LMS に保存された関連する目標 ID を返し，エラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o 集合に対するデータモデルバインディングはパ

	ックされたアレイとして表される．SCO が LMS が現在保持しているより大きいインデックス(n)を GetValue()要求で呼び出すと (アレイに 3 目標しかないときに要求は n 値が 5 を示す), LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 • **SetValue():** LMS は *cmi.objectives.n.id* を SetValue()要求で供給された値に設定し，エラーコードを“0” – No error に設定し“true”を返す o SetValue()の供給された値がデータモデル要素実装要件の要件を満たさないと，LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求にもとづていデータモデル要素の状態を変える事はない o 集合データモデル要素はシーケンシャル順に設定されるように要求される．SCO が目標をシーケンシャル順に設定しないと，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o SetValue()で供給された値が学習アテンプトにおいて配列の早い位置で既に使われた値(学習目標情報のセットの範囲内で一意でない)なら, LMS はエラーコードを 351 - General Set Failure に設定し"false"を返す. セクション 3.1.7.6: *SCORM Extension Error Conditions* を参照のこと. LMS は要求に基づくデータモデル要素の状態を変化させない o SetValue()要求の値が現在の値と同じであれば, LMS はエラーコードを 0 － No Error に設定し, “ true ”を返却する **追加動作要件:** SCO は新しい目標情報がインデックスリストにシーケンシャル順に挿入されることを確実にする責任がある. *cmi.objectives.n.id* は他のどんな目標情報よりも前に最初に設定されるように要求されている **例:** • GetValue(“cmi.objectives.0.id”) • SetValue(“cmi.objectives.0.id”,”obj1”)
Score	
cmi.objectives.n.score._children	*cmi.objectives.n.score._children* データモデル要素は，サポートされるデータモデル要素のリストを表す．このデータモデル要素は一般的にどのデータモデル要素が LMS にサポートされているか特定するのに SCO に利用される．返された文字列は，GetValue()および SetValue()要求に対してパラメータを動的に作成するのに SCO に利用されることがある **データモデル要素実装要件:** • **データタイプ**: 文字列 • **値空間:** ISO-10646-1 [5]

	• **フォーマット:** LMS にサポートされた親対話データモデル要素の全てのデータモデル要素のコンマで区切られたリスト．全てのデータモデル要素は LMS にサポートされるように要求されているので，文字列は以下のデータモデル要素を表す： o scaled o raw o min o max **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS はコンマで区切られた全てのデータモデル要素のリストを返す責任がある (上記データモデル要素実装要件参照) **SCO 動作要件:** • LMS はこのデータモデル要素を read-only.として実装する必要がある．SCO は *cmi.objectives.n.score._children* データモデル要素の値を検索することだけが許されている **API 実装要件:** • **GetValue():** LMS はコンマで区切られた LMS にサポートされるデータモデル要素のリストを返し(上記データモデル要素実装要件参照)，エラーコードを“0” – No error に設定する．データモデル要素の順序は重要でない．返された文字列はデータモデル要素実装要件で特定された要件を遵守する • **SetValue():** *cmi.objectives.n.score._children* を設定するために，SCO が SetValue()要求を呼び出すと，LMS はエラーコードを“404” – Data Model Element Is Read Only に設定し “false”を返す **例:** • GetValue(“cmi.objectives.0.score._children”)
cmi.objectives.n.score.scaled	*cmi.objectives.n.score.scaled* データモデル要素は，目標に対する学習者のパフォーマンスを反映する数字である．このデータモデル要素の値は－1 から 1 の範囲に合うようにスケールされる [1]. 測定に頼る SCO に関連する学習アクティビティに適用されたシーケンシング情報があれば，SCO は SCO の学習者セッションが終了する前にスコア情報が正確に LMS に送られたことを確実にしなければならない．そうでないと，シーケンシング情報を処理するとき，LMS は，SCO と関連する学習アクティビティの目標に対する目標測定として“unknown”値を利用する **データモデル要素実装要件:** • **データタイプ**: 実数(10,7)範囲(-1..1) • **値空間**: 有効桁数小数点以下 7 桁の実数．値の範囲は-1.0 から 1.0 • **フォーマット**: 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO は *cmi.objectives.n.score.scaled* を特定する責任がある．LMS は，SCO からレポートされない限り，目標の

	スケールされたスコアに判断が下せない．LMS が SCO に値が設定される前に検索要求(GetValue())を受け取ると，LMS は適切なエラーコードを設定する (API 実装要件参照) **シーケンシングインパクト:** • SCO が SCO の目標に対して *cmi.objectives.n.score.scaled* を設定しないと，SCO と関連する学習アクティビティの関連する目標に対する *Objective Measure Status* は false になる • SCO が SCO の目標に対して *cmi.objectives.n.score.scaled* を設定すると，SCO と関連する学習アクティビティの関連する目標に対する *Objective Measure Status* は true になり，SCO と関連する学習アクティビティの関連する目標に対する *Objective Normalized Measure* は score.scaled の値と等しくなる **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.objectives.n.score.scaled* データモデル要素の値を検索および保存することが許されている **API 実装要件:** • **GetValue():** LMS は，学習者に対して現在 LMS に保存された関連する *cmi.objectives.n.score.scaled* を返し，エラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o 集合に対するデータモデルバインディングはパックされたアレイとして表される．SCO が LMS が現在保持しているより大きいインデックス(n)を GetValue()要求で呼び出すと (アレイに 3 目標しかないときに要求は n 値が 5 を示す), LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o SCO が *cmi.objectives.n.score.scaled* を検索しようとし，データのレコードが作成されるが, scaled データモデル要素が SCO に設定されていないと，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返す • **SetValue():** LMS は *cmi.objectives.n.score.scaled* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータへ設定し，エラーコードを“0” – No error に設定し“true”を返す o SCO が *cmi.objectives.n.score.scaled* を実数でない値に設定しようとすると LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求にもとづていデータモデル要素の状態を変える事はない o SCO が *cmi.objectives.n.score.scaled* を実数であるが−1 から 1 の範囲にない値に設定しようとすると，LMS はエラーコードを“407” - Data Model Element Value Out Of Range に設定し

	“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない o *cmi.objectives.n.id* は他の目標情報よりも前に最初に設定されるように要求されているので，SCO が *cmi.objectives.n.score.scaled* を（識別子を設定する前に）設定しようとすると，LMS はエラーコードを“408” – Data Model Dependency Not Established に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない o 集合に対するデータモデルバインディングはパックされたアレイとして表される．SCO が現在保持されている目標の数より大きいインデックス(n)を SetValue()要求で呼び出すと，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 **例:** • GetValue(“cmi.objectives.0.score.scaled”) • SetValue(“cmi.objectives.0.score.scaled”,”0.750033”) • SetValue(“cmi.objectives.0.score.scaled”,”0.75”)
cmi.objectives.n.score.raw	*cmi.objectives.n.score.raw* データモデル要素は，目標に対して，学習者のパフォーマンスを最小値と最大値の間の範囲で反映する数字である[1] **データモデル要素実装要件:** • **データタイプ**: 実数(10,7) • **値空間**: 有効桁数小数点以下 7 桁の実数 • **フォーマット**: 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO は，この値が適切であれば，設定する責任がある．LMS は，SCO からレポートされない限り，目標の実スコアに判断が下せない．LMS が SCO に値が設定される前に検索要求(GetValue())を受け取ると，LMS は API 実装要件に従って動作する **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.objectives.n.score.raw* データモデル要素の値を検索および保存することが許されている．目標に対する実スコアはどのような方法で決定され計算されても良く，SCO によってコントロールされる **API 実装要件:** • **GetValue()**: LMS は，学習者に対して現在 LMS に保存された関連する *cmi.objectives.n.score.raw* を返し，エラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o 集合に対するデータモデルバインディングはパックされたアレイとして表される．SCO が LMS が現在保持しているより大きいインデックス(n)を GetValue()要求で呼び出すと (アレイに 3 目標しかないときに要求は n 値が 5 を示す),

	LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o SCO が *cmi.objectives.n.score.raw* を検索しようとし，データのレコードが作成されるが，raw データモデル要素が SCO に設定されていないと，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返す • **SetValue():** LMS は *cmi.objectives.n.score.raw* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータへ設定し，エラーコードを“0” – No error に設定し“true”を返す o SCO が *cmi.objectives.n.score.raw* を実数でない値に設定しようとすると LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求にもとづていデータモデル要素の状態を変える事はない o 集合に対するデータモデルバインディングはパックされたアレイとして表される．SCO が現在保持されている目標の数より大きいインデックス(n)を SetValue()要求で呼び出すと，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o *cmi.objectives.n.id* は他の目標情報よりも前に最初に設定されるように要求されているので，SCO が *cmi.objectives.n.score.raw* を（識別子を設定する前に）設定しようとすると，LMS はエラーコードを“408” – Data Model Dependency Not Established に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない **例:** • GetValue(“cmi.objectives.0.score.raw”) • SetValue(“cmi.objectives.0.score.raw”,”75.0033”) • SetValue(“cmi.objectives.0.score.raw”,”0.75”)
cmi.objectives.n.score.min	*cmi.objectives.n.score.min* データモデル要素は，目標に対する実スコアの範囲の最小値である[1] **データモデル要素実装要件:** • **データタイプ:** 実数(10,7) • **値空間:** 有効桁数小数点以下 7 桁の実数 • **フォーマット:** 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO は，この値が適切であれば，設定する責任がある．LMS は，SCO からレポートされない限り，目標の最小スコアに判断が下せない．LMS が SCO に値が設定される前に検索要求(GetValue())を受け取ると，LMS は適切なエラーコードを設定する（API 実装要件参照） **SCO 動作要件:**

	• LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.objectives.n.score.min* データモデル要素の値を検索および保存することが許されている．最小スコアは SCO に決定される **API 実装要件:** • **GetValue():** LMS は，学習者に対して現在 LMS に保存された関連する *cmi.objectives.n.score.min* を返し，エラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o 集合に対するデータモデルバインディングはパックされたアレイとして表される．SCO が LMS が現在保持しているより大きいインデックス(n)を GetValue()要求で呼び出すと (アレイに 3 目標しかないときに要求は n 値が 5 を示す), LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o SCO が *cmi.objectives.n.score.min* を検索しようとし，データのレコードが作成されるが，min データモデル要素が SCO に設定されていないと，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返す • **SetValue():** LMS は *cmi.objectives.n.score.min* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータへ設定し，エラーコードを“0” – No error に設定し“true”を返す o SCO が *cmi.objectives.n.score.min* を実数でない値に設定しようとすると LMS はエラーコードを “406” – Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求にもとづていデータモデル要素の状態を変える事はない o 集合に対するデータモデルバインディングはパックされたアレイとして表される．SCO が現在保持されている目標の数より大きいインデックス(n)を SetValue()要求で呼び出すと，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o *cmi.objectives.n.id* は他の目標情報よりも前に最初に設定されるように要求されているので，SCO が *cmi.objectives.n.score.min* を（識別子を設定する前に）設定しようとすると，LMS はエラーコードを“408” – Data Model Dependency Not Established に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない **例:** • GetValue(“cmi.objectives.0.score.min”) • SetValue(“cmi.objectives.0.score.min”,”1.0”) • SetValue(“cmi.objectives.0.score.min”,”500”)
cmi.objectives.n.score.max	*cmi.objectives.n.score.max* データモデル要素は，目標に対する実ス

	コアの範囲の最大値である[1] **データモデル要素実装要件:** • **データタイプ:** 実数(10,7) • **値空間:** 有効桁数小数点以下 7 桁の実数 • **フォーマット:** 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO は，この値が適切であれば，設定する責任がある．LMS は，SCO からレポートされない限り，目標の最大スコアに判断が下せない．LMS が SCO に値が設定される前に検索要求(GetValue())を受け取ると，LMS は適切なエラーコードを設定する（API 実装要件参照） **SCO 動作要件:** LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.objectives.n.score.max* データモデル要素の値を検索および保存することが許されている．最大スコアは SCO に決定される **API 実装要件:** • **GetValue():** LMS は，学習者に対して現在 LMS に保存された関連する *cmi.objectives.n.score.max* を返し，エラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o 集合に対するデータモデルバインディングはパックされたアレイとして表される．SCO が LMS が現在保持しているより大きいインデックス(n)を GetValue()要求で呼び出すと (アレイに 3 目標しかないときに要求は n 値が 5 を示す), LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o SCO が *cmi.objectives.n.score.max* を検索しようとし，データのレコードが作成されるが，max データモデル要素が SCO に設定されていないと，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返す • **SetValue():** LMS は *cmi.objectives.n.score.max* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータへ設定し，エラーコードを“0” – No error に設定し“true”を返す o SCO が *cmi.objectives.n.score.max* を実数でない値に設定しようとすると LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求にもとづていデータモデル要素の状態を変える事はない o 集合に対するデータモデルバインディングはパックされたアレイとして表される．SCO が現在保持されている目標の数より大きいインデックス(n)を SetValue()要求で呼び出すと，LMS はエラーコードを“351” – General Set Failure に設定

	し，“false”を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o *cmi.objectives.n.id* は他の目標情報よりも前に最初に設定されるように要求されているので，SCO が *cmi.objectives.n.score.max* を（識別子を設定する前に）設定しようとすると，LMS はエラーコードを“408” – Data Model Dependency Not Established に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない **例:** • GetValue(“cmi.objectives.0.score.max”) • SetValue(“cmi.objectives.0.score.max”,”1.0”) • SetValue(“cmi.objectives.0.score.max”,”500”)
cmi.objectives.n.success_status	*cmi.objectives.n.success_status* データモデル要素は学習者が目標をマスターしたかどうかを示す[1]．SCO がどのように目標に対して *cmi.objectives.n.success_status* を決定するかは SCORM の対象範囲外である．SCO はいくつかの根拠によりこの決定を下すことができる：目標にマッピングする対話の比率，テストやクイズの合計スコア，目標に基いて，マスタースコアに対比される等々．この値は，SCO 開発者に決定されたように，SCO に対する全体的な success status を示す． 目標ステータスに依存し，SCO と関連する学習アクティビティに適用されたシーケンシング情報があれば，SCO は，SCO の学習者セッション終了の前に目標情報が正確に LMS に送られたことを確実にしなければならない．そうでないと，シーケンシング情報を処理するとき，LMS は，SCO と関連する学習アクティビティの目標に対する目標ステータスとして“unknown”値を利用する **データモデル要素実装要件:** • **データタイプ**: 状態 (passed, failed, unknown) • **値空間**: IEEE ドラフトは 3 つの状態値を定義する．SCORM はこれらの値を以下の制限された語彙トークンにバインドする： o “passed”: 学習者が SCO をパスした[1]．必要な数の目標がマスターされたもしくは必要なスコアが達成されたことを示す o “failed”: 学習者が SCO を失敗した[1]．学習者が必要な数の目標をマスターしなかったもしくは要求されたスコアが達成されなかったことを示す o “unknown”: 主張ができない[1]．成功ステータスを示す主張が適用できないことを示す • **フォーマット**: このデータモデル要素のフォーマットは上記の 3 つの語彙トークンのうちの一つ (“passed”, “failed”, “unknown”) **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • 通常 SCO は自分の *cmi.objectives.n.success_status* を LMS にレポートするが，SCORM には SCO が *cmi.objectives.n.success_status* を設定することを義務付ける要件はない．*cmi.objectives.n.success_status* が SCO にレポートされないと，LMS は目標成功ステータスの値とし

	てデフォルトの“unknown”を利用する **シーケンシングインパクト:** • SCO が SCO の目標に対して *cmi.objectives.n.success_status* を“unknown”に設定すると SCO に関連する学習アクティビティの目標に対する *Objective Progress Status* は false である． • SCO が SCO の目標に対して *cmi.objectives.n.success_status* を“passed”に設定すると，SCO と関連する学習アクティビティの関連する目標に対する *Objective Progress Status* は true になり，SCO と関連する学習アクティビティの関連する目標に対する *Objective Satisfied Status* は true となる • SCO が SCO の目標に対して *cmi.objectives.n.success_status* を“failed”に設定すると，SCO と関連する学習アクティビティの関連する目標に対する *Objective Progress Status* は true になり，SCO と関連する学習アクティビティの関連する目標に対する *Objective Satisfied Status* は false となる **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.objectives.n.success_status* データモデル要素の値を検索および保存することが許されている． **API 実装要件:** • **GetValue():** LMS は，学習者に対して現在 LMS に保存された関連する *cmi.objectives.n.success_status* を返し，エラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する．*cmi.objectives.n.success_status* が設定されるまで，このデータモデル要素のデフォルト値は“unknown”である o SCO が *cmi.objectives.n.success_status* の取得を試み, かつデータのレコードが作られ *success_status* データモデル要素が SCO によって設定されず LMS によっても決定されなかったら, LMS はデフォルト値の"unknown"を返し, エラーコードを 0 - No Error に設定する. o 集合に対するデータモデルバインディングはパックされたアレイとして表される．SCO が LMS が現在保持しているより大きいインデックス(n)を GetValue()要求で呼び出すと (アレイに 3 目標しかないときに要求は n 値が 5 を示す), LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 • **SetValue():** SCO が *cmi.objectives.n.success_status* を設定するために SetValue()要求を呼び出し，供給された値がデータモデル要素実装要件で定義された要件に満たないと，LMS は *cmi.objectives.n.success_status* に対して供給された値を保存し，エラーコードを“0” – No error に設定し“true”を返す o SCO が *cmi.objectives.n.success_status* を設定する要求を呼び出し，値が上記の制限された語彙トークンの一つでないと，LMS はエラーコードを“406” – Data Model Element Type Mismatch に設

	定し“false”を返す．LMS はこの要求にもとづていデータモデル要素の状態を変える事はない o 集合に対するデータモデルバインディングはパックされたアレイとして表される．SCO が現在保持されている目標の数より大きいインデックス(n)を SetValue()要求で呼び出すと，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o *cmi.objectives.n.id* は他の目標情報よりも前に最初に設定されるように要求されているので，SCO が *cmi.objectives.n.success_status* を（識別子を設定する前に）設定しようとすると，LMS はエラーコードを“408” – Data Model Dependency Not Established に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない **例:** • GetValue(“cmi.objectives.n.success_status”) • SetValue(“cmi.objectives.n.success_status”,”passed”)
cmi.objectives.n.completion_status	*cmi.objectives.n.completion_status* データモデル要素は学習者が関連する目標を完了したかどうかを示す[1]．SCO がどのように目標に対して *cmi.objectives.n.completion_status* を決定するかは SCORM の対象範囲外である．例えば，SCO はこの決定を完了した目標と関連するインタラクションの数に基づいて行うことができる． *cmi.objectives.n.completion_status* の決定は SCO にコントロールされ管理されるので，LMS は SCO が完了したと示唆することができない．*cmi.objectives.n.completion_status* が SCO にレポートされないと，LMS は *cmi.objectives.n.completion_status* は“unknown”であるという事実に頼るほかない **データモデル要素実装要件:** • **データタイプ**: 状態 (completed, incomplete, not_attempted, unknown) • **値空間**: IEEE ドラフトは 4 つの状態値を定義する．SCORM はこれらの値を以下の制限された語彙トークンにバインドする o “completed”: 学習者は関連するオブジェクトに対して SCO を完了したと考慮されるほど充分に経験した[1]．どのように完了を決定するかは SCO にコントロールおよび管理される o “incomplete”: 学習者は関連するオブジェクトに対して SCO を完了したと考慮されるほど充分に経験していない[1]．どのように完了を決定するかは SCO にコントロールおよび管理される o “not attempted”: 学習者は SCO を経験しているが，関連するオブジェクトがアテンプトされていない[1]．SCO はオブジェクトがアテンプトされたかどうか決定する責任がある o “unknown”: 主張ができない[1]．これは，完了ステータスを示す主張が適用できないことを示す • **フォーマット**: このデータモデル要素のフォーマットは，上記の制限された語彙トークンのうちの一つである

	(“completed”, “incomplete”, “not attempted”, “unknown”) **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • 通常 SCO は自分の目標完了ステータスを LMS にレポートするが，SCORM には SCO が *cmi.objectives.n.completion_status* を設定することを義務付ける要件はない．SCO が *cmi.objectives.n.completion_status* を設定しないと LMS は *cmi.objectives.n.completion_status* を “unknown”とした扱う **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.objectives.n.completion_status* データモデル要素の値を検索および保存することが許されている． **API 実装要件:** • **GetValue():** LMS は，学習者に対して現在 LMS に保存された関連する *cmi.objectives.n.completion_status* を返し，エラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する．*cmi.objectives.n.completion_status* が設定されるまで，このデータモデル要素のデフォルト値は“unknown” である o SCO が *cmi.objectives.n.completion_status* の取得を試み, かつデータのレコードが作られ *completion_status* データモデル要素が SCO によって設定されず LMS によっても決定されなかったら, LMS はデフォルト値の"unknown"を返し, エラーコードを 0 - No Error に設定する. o 集合に対するデータモデルバインディングはパックされたアレイとして表される．SCO が LMS が現在保持しているより大きいインデックス(n)を GetValue()要求で呼び出すと (アレイに 3 目標しかないときに要求は n 値が 5 を示す), LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 • **SetValue():** データ依存性が満たされると（目標に対する識別子があるインデックス *n* に以前に設定されており，供給される値がデータモデル要素実装要件を満たすと），LMS は *cmi.objectives.n.completion_status* データモデル要素を供給された値に設定しエラーコードを“0” – No error に設定し“true”を返す o SCO が *cmi.objectives.n.completion_status* を設定する要求を呼び出し，値が上記の制限された語彙トークンの一つでないと，LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求にもとづいていデータモデル要素の状態を変える事はない o 集合に対するデータモデルバインディングはパックされたアレイとして表される．SCO が現在保持されている目標の数より大きいインデック

	ス(n)を SetValue()要求で呼び出すと，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o *cmi.objectives.n.id* は他の目標情報よりも前に最初に設定されるように要求されているので，SCO が *cmi.objectives.n.completion_status* を（識別子を設定する前に）設定しようとすると，LMS はエラーコードを“408” – Data Model Dependency Not Established に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない **例:** • GetValue(“cmi.objectives.0.completion_status”) • SetValue(“cmi.objectives.0.completion_status”,”incomplete”)
cmi.objectives.n.progress_measure	*cmi.objectives.n.progress_measure* データモデル要素は学習者が関連する学習目標に向けて行った達成の度合いを示す [1]. SCO がどのように学習目標の *cmi.objectives.n.progress_measure* を決めるかは SCORM 規格の範囲外である. 例えば, SCO は, 完了されている目的と関連したいくつかのインタラクションをこの決定の根拠とすることができるだろう. *cmi.objectives.n.progress_measure* の決定が SCO によってコントロールされ管理されるので, LMS はどんな形であれ進捗度の値に意味を与えることができない. **データモデル要素実装要件:** • **データタイプ**: 実数(10,7)範囲(0..1) • **値空間**: 有効桁数小数点以下 7 桁の実数．値は 0.0 から 1.0 の範囲にある • **フォーマット**: 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO は，*cmi.objectives.n.progress_measure* を決定する責任がある．LMS は，SCO からレポートされない限り，*cmi.objectives.n.progress_measure* に判断が下せない．LMS が値が設定される前に検索要求(GetValue())を受け取ると，LMS は適切なエラーコードを設定する（API 実装要件参照） **SCO 動作要件:** LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.objectives.n.progress_measure* データモデル要素の値を検索および保存することが許されている． **API 実装要件:** • **GetValue()**: LMS は，学習者に対して現在 LMS に保存された関連する *cmi.objectives.n.score.max* を返し，エラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o 集合に対するデータモデルバインディングは詰まった配列として表現される. index (n)が LMS が現在維持する数よりも大きな数である所(例えば, 3 つの学習目標だけが配列にある時, 要求が

	n の値として 5 を示した)で SCO が GetValue()要求を行ったら, LMS はエラーコードを 301 - General Get Failure に設定し空文字列("")を返す. 要求の処理における推奨に関してはセクション 3.1.7.6: *SCORM Extension Error Conditions* を参照のこと. o SCO が値が設定される前に *cmi.objectives.n.progress_measure* を得る要求を呼び出すと，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返す • **SetValue():** LMS は *cmi.objectives.n.progress_measure* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータへ設定し，エラーコードを“0” – No error に設定し“true”を返す o SCO が *cmi.objectives.n.progress_measure* を実数でない値に設定しようとすると LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求にもとづていデータモデル要素の状態を変える事はない o 集合に対するデータモデルバインディングは詰まった配列として表現される. 供給された index (n)が現在保持される学習目標の数よりも大きい数である所で SCO が SetValue()要求を行ったら, LMS はエラーコードを 351 - General Set Failure に設定し"false"を返す. セクション 3.1.7.6: *SCORM Extension Error Conditions* を参照のこと. o SCO が *cmi.objectives.n.progress_measure* を実数であるが 0 から 1 の範囲にない値に設定しようとすると，LMS はエラーコードを“407” - Data Model Element Value Out Of Range に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない o *cmi.objectives.n.id* が他のどんな学習目標情報よりも先に設定されることが要求されるので, SCO が *cmi.objectives.n.progress_measure* を設定しようと試みたら(識別子の設定よりも前), LMS はエラーコードを 408 - Data Model Dependency Not Established に設定し"false"を返す. LMS は要求に基づくデータモデル要素の状態を変えない. **例:** • GetValue(“cmi.objectives.0.progress_measure”) • SetValue(“cmi.objectives.0.progress_measure”,”0.75”) • SetValue(“cmi.objectives.0.progress_measure”,”1.0”)
cmi.objectives.n.description	*cmi.objectives.n.description* データモデル要素は目的についての短い情報の記述である **データモデル要素実装要件:** • **データタイプ:** localized_string_type (SPM: 250) • **値空間:** ローカライズされた情報を持つ文字列（ISO-10646-1 で定義される） • **フォーマット:** より詳細はセクション 4.1.1.7: *データタイプ*参照

	LMS 動作要件: • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • この値は現在 SCO に初期化される．SCORM はこの値を初期化する他の方法を定義しない．LMS はこのデータモデル要素に対する初期値に関するいかなる仮定もしない．SCO に実際の目標記述が設定される前に GetValue() 要求がなされると，LMS は API 実装要件に従って動作する **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.objectives.n.description* データモデル要素の値を検索および保存することが許されている • SetValue()要求中，SCO は，デリミターはオプションであることに留意すべきである．デリミターが文字列の 1 部として提供されなければ，LMS はデフォルト言語は “en”（英語）であると仮定する • GetValue()要求中，SCO は，デリミターは LMS に返される文字列の 1 部である(LMS の実装による)ことがあることに留意すべきである．SCO が LMS に返された文字列と何をするかは SCO の実装による． **API 動作要件:** • **GetValue():** LMS は，学習者に対して現在 LMS に保存された関連する目標記述を返し，エラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する． o 集合に対するデータモデルバインディングはパックされたアレイとして表される．SCO が LMS が現在保持しているより大きいインデックス(n)を GetValue()要求で呼び出すと (アレイに 3 目標しかないときに要求は n 値が 5 を示す), LMS はエラーコードを“301” – General Get Failure に設定し，空の文字列(“”)を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o SCO が *cmi.objectives.n.description* を検索しようとし，データのレコードが作成されるが, description データモデル要素が SCO に設定されていないと，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返す • **SetValue():** LMS は *cmi.objectives.n.description* を SetValue()要求で供給された値に設定しエラーコードを “0” – No error に設定し“true”を返す o SetValue()の供給された値がデータモデル要素実装要件の要件に満たないと，LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求にもとづいていデータモデル要素の状態を変える事はない o 集合に対するデータモデルバインディングはパックされたアレイとして表される．SCO が現在保持されている目標の数より大きいインデック

	ス(n)を SetValue()要求で呼び出すと，LMS はエラーコードを“351” – General Set Failure に設定し，“false”を返す．より詳細はセクション 3.1.7.6: *SCORM 拡張エラー条件*参照 o *cmi.objectives.n.id* は他の目標情報よりも前に最初に設定されるように要求されているので，SCO が *cmi.objectives.n.success_status* を（識別子を設定する前に）設定しようとすると，LMS はエラーコードを“408” – Data Model Dependency Not Established に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない **例:** • GetValue(“cmi.objectives.0.description”) • SetValue(“cmi.objectives.0.description”,”この単位の完了後，学習者はりんごとオレンジを区別することができるようになる”)

4.2.18. Progress Measure

cmi.progress_measure データモデル要素は，学習者が SCO 完了に向けて達成した進捗状況を測定する [1].

表 4.2.18a: Completion Status と進捗測定の関係

cmi.progress_measure	cmi.completion_status
0	“not attempted”
1	“completed”
0 > value < 1	“incomplete” (一般的に, `cmi.completion_threshold` が定義され，`cmi.progress_measure` >= `cmi.completion_threshold` でない限り)

表 4.2.1.8a に，cmi.progress_measure 値の cmi.completion_status 値へのマッピングングを示す．SCO は，進捗測定を決定するための要件を定義する．これは SCO に関連する目標の完了数，学習者に提示されるページ数等に基づいている

表 4.2.18b: Progress Measure データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	説明
cmi.progress_measure	**データモデル要素実装要件:** • **データタイプ**: 実数(10,7)範囲(0..1) • **値空間**: 有効桁数小数点以下 7 桁の実数．値は 0.0 から 1.0 の範囲にある • **フォーマット**: 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO は，*cmi.progress_measure* を決定する責任がある．LMS は，SCO からレポートされない限り，*cmi.progress_measure* に判断が下せない．LMS が値が設定される前に検索要求(GetValue())を受け取ると，LMS は適切なエラーコードを設定する（API 実装要件参照） • a *cmi.completion_threshold* が定義され，SCO が *cmi.progress_measure* をレポートすると，LMS は，セクション 4.2.4.1: *Completion Status Determination* で定義された要件に基づいて，現在 cmi.completion_status に保存された値を上書きする **シーケンシングインパクト:** • 現時点で，*cmi.progress_measure* から IMS Simple Sequencing Tracking Model 要素への直接のマッピングはない．シーケンシング動作への直接のマッピングもない． • *cmi.completion_threshold* が定義され *cmi.progress_measure* が設定されると，*cmi.completion_status* データモデル要素の決定が影響されることがある．*cmi.completion_status* はある特定のシーケンシング動作に影響を与える

	SCO 動作要件: • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.progress_measure* データモデル要素の値を検索および保存することが許されている． **API 動作要件:** • **GetValue():** LMS は，学習者に対して現在 LMS に保存された関連する *cmi.progress_measure* を返し，エラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する． o SCO が値が設定される前に *cmi.progress_measure* を得ようと要求を呼び出すと，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返す • **SetValue():** LMS は *cmi.progress_measure* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータへ設定し，エラーコードを“0” – No error に設定し“true”を返す o SCO が *cmi.progress_measure* を実数でない値に設定しようとすると LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求にもとづいていデータモデル要素の状態を変える事はない o SCO が *cmi.progress_measure* を実数であるが 0 から 1 の範囲にない値に設定しようとすると，LMS はエラーコードを“407” - Data Model Element Value Out Of Range に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない **例:** • GetValue(“cmi.progress_measure”) • SetValue(“cmi.progress_measure”,”0.75”) • SetValue(“cmi.progress_measure”,”1.0”)

4.2.19. Scaled Passing Score

cmi.scaled_passing_score データモデル要素は，SCO の終了に要求される調整された合格スコアである．このデータモデル要素の値は－1 から 1 の範囲に収まるように調整されている[1]．値は SCO に対する調整された合格スコアを示す．

表 4.2.19a: Scaled Passing Score データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.scaled_passing_score	**データモデル要素実装要件:** • **データタイプ:** 実数(10,7)範囲(-1..1) • **値空間:** -1.0 <= scaled_passing_score <= 1.0 • **フォーマット:** 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない. • LMS はデータモデル要素を初期化する責任を持つ. • SCO を参照する<imscp:item>要素の<imsss:primaryObjective>要素に関連する IMS シンプルシーケンシング名前空間属性 imsss:satisfiedByMeasure が true であるならば, SCO リソースを参照する<imscp:item>要素の<imsss:primaryObjective>要素に関連する<imsss:minNormalizedMeasure>要素の値は, このデータモデル要素を初期化して使われる. • SCO を参照する<imscp:item>要素の<imsss:primaryObjective>要素に関連する IMS シンプルシーケンシング名前空間属性 imsss:satisfiedByMeasure が true であり, かつ SCO リソースを参照する<imscp:item>要素の<imsss:primaryObjective>要素に関連する<imsss:minNormalizedMeasure>要素に値が供給されないならば, LMS はこのデータモデル要素を 1.0 に初期化する (デフォルト値). • SCO を参照する<imscp:item>要素の<imsss:primaryObjective>要素に関連する IMS シンプルシーケンシング名前空間属性 imsss:satisfiedByMeasure が false であるならば, LMS は scaled passing score に対するいかなる仮定も行わない. この場合, *cmi.scaled_passing_score* の値は初期化されない. **SCO Behavior Requirements:** • このデータモデル要素は LMS に read-only として実装される必要がある．SCO は *cmi.scaled_passing_score* データモデル要素の値を検索することだけが許されている． **API Implementation Requirements:** • **GetValue():** LMS は，*cmi.scaled_passing_score* データモデル要素に対して保存されたを返し，エラーコードを “0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する． o マニフェストに定義されたスケールされた合格スコアがなく，SCO がこのデータモデル要素に対する値を要求すると，LMS はエラーコードを “403” Data Model Element Value Not Initialized に

	設定し空の文字列(“”)を返す • **SetValue():** *cmi.scaled_passing_score* を設定するのに，SCO が SetValue()要求を呼び出すと，LMS はエラーコードを“404” – Data Model Element Is Read Only に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない **例:** • GetValue(“cmi.scaled_passing_score”)

4.2.20. Score

score データモデル要素は，SCO に対する学習者のスコアである．Score データモデル要素は 4 つのサブ要素に区分される：

- cmi.score.scaled: scaled データモデル要素は学習者のパフォーマンスの数字を反映する数字である．このデータモデル要素は−1.0 から 1.0 の範囲に収まるようにスケールされている [1]
- cmi.score.raw: raw データモデル要素は，最小値と最大値で仕切られた範囲と関連した学習者のパフォーマンスを反映する数字である[1]
- cmi.score.min: min データモデル要素は，raw スコアに対する範囲内での最小値である [1]
- cmi.score.max: max データモデル要素は，raw スコアに対する範囲内での最小値である [1]

表 4.2.20a: Score データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.score._children	*cmi.score._children* データモデル要素は，サポートされるデータモデル要素のリストを表す．このデータモデル要素は一般的にどのデータモデル要素が LMS にサポートされているか特定するのに SCO に利用される．返された文字列は，GetValue()および SetValue()要求に対してパラメータを動的に作成するのに SCO に利用されることがある **データモデル要素実装要件:** • **データタイプ**: 文字列 • **値空間:** ISO-10646-1 [5] • **フォーマット:** LMS にサポートされた親対話データモデル要素の全てのデータモデル要素のコンマで区切られたリスト．全てのデータモデル要素は LMS にサポートされるように要求されているので，文字列は以下のデータモデル要素を表す： o scaled o raw o min o max **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS はコンマで区切られた全てのデータモデル要素のリストを返す責任がある (上記データモデル要素実装要件参照) **SCO 動作要件:** • LMS はこのデータモデル要素を read-only.として実装する必要がある．SCO は *cmi.score._children* データモデル要素の値を検索することだけが許されている **API 実装要件:** • **GetValue():** LMS はコンマで区切られた LMS にサポートされるデータモデル要素のリストを返し(上記データモデル要素実装要件参照), エラーコードを“0” – No error に設定する．データモデル要素の順序は重要でない．返された文字列はデータモデル要素実装要件で特定された要件を遵守する • **SetValue():** *cmi.score_children* を設定するために，SCO が SetValue()要求を呼び出すと，LMS はエラーコードを“404” – Data Model Element Is Read Only に設定し “false”を返す **例:**

	• GetValue("cmi.score._children")
cmi.score.scaled	**データモデル要素実装要件:** • **データタイプ:** 実数(10,7)範囲(-1..1) • **値空間:** 有効桁数小数点以下 7 桁の実数．値の範囲は-1.0 から 1.0 • **フォーマット:** 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO はスケールされたスコアを決定する責任がある．LMS は，SCO からレポートされない限り，スケールされたスコアに判断が下せない．LMS が SCO に値が設定される前に検索要求(GetValue())を受け取ると，LMS は適切なエラーコードを設定する (API 実装要件参照) **シーケンシングインパクト:** *cmi.objectives.n.success_status* および *cmi.objectives.n.score.scaled* からのシーケンシングインパクトが考慮された後に以下が発生する： • SCO が *cmi.score.scaled* を設定しないと，SCO と関連する学習アクティビティの主要目標に対する *Objective Measure Status* は false になる • SCO が *cmi.score.scaled* を設定すると，SCO と関連する学習アクティビティの関連する目標に対する *Objective Measure Status* は true になり，SCO と関連する学習アクティビティの関連する目標に対する *Objective Normalized Measure* は score.scaled の値と等しくなる **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.score.scaled* データモデル要素の値を検索および保存することが許されている • 測定に依存し，SCO と関連する学習アクティビティに適用されたシーケンシング情報があれば，SCO は，SCO の学習者セッション終了の前にスコア情報が正確に LMS に送られたことを確実にしなければならない．そうでないと，シーケンシング情報を処理するとき，LMS は，SCO と関連する学習アクティビティの主要目標に対する目標測定として"unknown"値を利用する **API 実装要件:** • **GetValue():** LMS は，学習者に対して現在 LMS に保存された関連する *cmi.score.scaled* を返し，エラーコードを"0" – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o SCO が，SCO に値が設定される前に，*cmi.score.scaled* を得る要素を呼び出すと，LMS はエラーコードを"403" Data Model Element Value Not Initialized に設定し空の文字列("")を返す • **SetValue():** LMS は *cmi.score.scaled* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータへ設定し，エラーコードを"0" – No error に設定し"true"を返す o SCO が *cmi.score.scaled* を実数でない値に設定しようとすると LMS はエラーコードを"406" – Data Model Element Type Mismatch に設定し"false"を返す．LMS はこの要求にもとづいていデータモデル要素の状態を変える事はない o SCO が *cmi.score.scaled* を実数であるが−1 から 1 の範囲にない値に設定しようとすると，LMS はエラーコードを"407" - Data Model Element Value Out Of Range に設定し"false"を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない **例:**

	• GetValue(“cmi.score.scaled”) • SetValue(“cmi.score.scaled”,”0.750033”) • SetValue(“cmi.score.scaled”,”0.75”)
cmi.score.raw	**データモデル要素実装要件:** • **データタイプ**: 実数(10,7) • **値空間**: 有効桁数小数点以下 7 桁の実数 • **フォーマット**: 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO は，この値が適切であれば，設定する責任がある．LMS は，SCO からレポートされない限り，実スコアに判断が下せない．LMS が SCO に値が設定される前に検索要求(GetValue())を受け取ると，LMS は適切なレラーコードを設定する（API 実装要件参照） **SCO 動作要件:** • LMS はこのデータモデル要素を LMS に read/write として実装する必要がある．SCO は *cmi.score.raw* データモデル要素の値を検索および保存することが許されている． • 目標に対する実スコアは SCO に分かればどのような方法で決定され計算されても良い．例えば，目的完了の比率を反映しても良いし，マルティプルチョイスの実スコアでも良く，SCO で後続の質問があるレスポンスに関する正しい最初のレスポンスの数でも良い **API 実装要件:** • **GetValue()**: LMS は，学習者に対して現在 LMS に保存された関連する *cmi.score.raw* を返し，エラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o SCO が，SCO に値が設定される前に，*cmi.score.raw* を得る要素を呼び出すと，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返す • **SetValue()**: LMS は *cmi.score.raw* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータへ設定し，エラーコードを“0” – No error に設定し“true”を返す o SCO が *cmi.score.raw* を実数でない値に設定しようとすると LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求にもとづいていデータモデル要素の状態を変える事はない **例:** • GetValue(“cmi.score.raw”) • SetValue(“cmi.score.raw”,”75.0033”) • SetValue(“cmi.score.raw”,”0.75”)
cmi.score.max	**データモデル要素実装要件:** • **データタイプ**: 実数(10,7) • **値空間**: 有効桁数小数点以下 7 桁の実数 • **フォーマット**: 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO は，この値が適切であれば，設定する責任がある．LMS は，SCO からレポートされない限り，最大スコアに判断が下せない．LMS が SCO に値が設定される前に検索要求(GetValue())を受け取ると，LMS は適切なレラーコードを設定する（API 実装要件参照） **SCO 動作要件:**

	• LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.score.max* データモデル要素の値を検索および保存することが許されている．最大スコアは SCO に決定される **API 実装要件:** • **GetValue():** LMS は，学習者に対して現在 LMS に保存された関連する *cmi.score.max* を返し，エラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o SCO が，SCO に値が設定される前に，*cmi.score.max* を得る要素を呼び出すと，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返す • **SetValue():** LMS は *cmi.score.max* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータへ設定し，エラーコードを“0” – No error に設定し“true”を返す o SCO が *cmi.score.max* を実数でない値に設定しようとすると LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求にもとづいていデータモデル要素の状態を変える事はない **例:** • GetValue(“cmi.score.max”) • SetValue(“cmi.score.max”,”1.0”) • SetValue(“cmi.score.max”,”500”)
cmi.score.min	**データモデル要素実装要件:** • **データタイプ:** 実数(10,7) • **値空間:** 有効桁数小数点以下 7 桁の実数 • **フォーマット:** 詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • SCO は，この値が適切であれば，設定する責任がある．LMS は，SCO からレポートされない限り，最小スコアに判断が下せない．LMS が SCO に値が設定される前に検索要求(GetValue())を受け取ると，LMS は適切なレラーコードを設定する（API 実装要件参照） **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.score.min* データモデル要素の値を検索および保存することが許されている．最小スコアは SCO に決定される **API 実装要件:** • **GetValue():** LMS は，学習者に対して現在 LMS に保存された関連する *cmi.score.min* を返し，エラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する o SCO が，SCO に値が設定される前に，*cmi.score.min* を得る要素を呼び出すと，LMS はエラーコードを“403” Data Model Element Value Not Initialized に設定し空の文字列(“”)を返す • **SetValue():** LMS は *cmi.score.min* データモデル要素を SetValue()コールの parameter_2 として通ったパラメータへ設定し，エラーコードを“0” – No error に設定し“true”を返す o SCO が *cmi.score.min* を実数でない値に設定しようとすると LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求にもとづいていデータモデル要素の状態を変える事はない

	例: • GetValue(“cmi.score.min”) • SetValue(“cmi.score.min”,”1.0”) • SetValue(“cmi.score.min”,”500”)

4.2.21. Session Time

cmi.session_time データモデル要素は，SCO に対する，現在の学習者セッションで学習者が費やした時間の量である．学習者セッションが進行していないと，セッションタイムは，この SCO に対して，最後の学習者セッションで学習者が費やした時間になる[1]．

SCO が，cmi.session_time を追跡するなら，SCO は，cmi.session_time の値と意味を決定する．例えば：

1. SCO は，SCO がメディアセグメントを初期化した後まで，レコーディングセッションを始めてはいけない
2. 学習者が学習者セッションで休憩をとれば，休憩時間は，報告される cmi.session_time に含まれることも含まれないこともある[1]

SCO は，常に追跡する役割がある．これは，そのような時間を記録するために，内部時計を実装することも含む．

表 4.2.21a: Session Time データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	説明
cmi.session_time	***データモデル要素:*** • **データタイプ:** timeinterval (second,10,2) – 精度 0.01 秒の時間インターバル • **フォーマット:** より詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須は LMS に write-only として実装しなくてはならない • このデータモデル要素はに LMS に write-only として実装されるので，LMS はこのデータモデル要素を初期化する責任はない．この値を管理するのは SCO の責任である．LMS の責任は，このデータモデル要素に SetValue()コールを受け入れ，*cmi.total_time* の累計を実施するだけである • SCO はこのデータモデル要素に対して値を設定するように要求されないので(セッションタイムを追跡するよう要求されない)，LMS は，LMS が SCO を起動してからのセッションタイムを追跡する．SCO が異なったセッションタイムをレポートすると，LMS は，LMS に計測されたセッションタイムではなく，SCO にレポートされたセッションタイムを利用する **SCO 動作要件:** • LMS はこのデータモデル要素を write-only として実装する必要がある．SCO は *cmi.session_time* データモデル要素の値を保存することだけが許されている • SCO は，一回の交信セッションで，複数セットのセッションタイム(*cmi.session_time*)を実施することを許されている．SCO 開発者は，SCO が Terminate("")を出すかもしくはユーザーが離れて行くと，LMS は最後の *cmi.session_time* を取得し，この時間を *cmi.total_time* に累積するという事実に留意すべきである **API 実装要件:** • **GetValue():** SCO が *cmi.session_time* を得る要求を呼び出すと，LMS はエラーコードを"405" – Data Model Element Is Write Only に設定し

	空の文字列(“”)を返す • **SetValue():** この要求は *cmi.session_time* を供給された値に設定する．供給された値は上記で定義されたデータモデル要素実装要件を満たすべきである．供給された値が正しければ，LMS は値を設定し“true”を返しエラーコードを“0” - No error に設定する o 供給された値が上記のデータモデル要素実装要件を満たさないと，LMS はレラーコードを“406” - Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない **追加動作要件:** • この値は学習セッションに対して費やされた時間を追跡する．LMS は，この値を *cmi.total_time* を決定するのに利用する．SCO は，一回の学習者セッションで，*cmi.session_time* の SetValue() を複数実施することができる．SCO が Terminate(“”)を出すかもしくはユーザーが離れて行くと，LMS は最後の *cmi.session_time* を取得しこの時間を *cmi.total_time* に累計する．同一の学習者アテンプトで SCO の後続の起動があり *cmi.total_time* に対する GetValue()があり次第，LMS は累積時間を返す．LMS は SetValue()要求を通して送られた複数のセッションタイムは累積しない．*cmi.session_time* に対する複数の SetValue()コールがなされると，LMS は内部で様々なセッションタイムを追跡し，*cmi.session_time* を保持するときに最後の値だけを利用する **例:** • SetValue(“cmi.session_time”,”PT1H5M”)

4.2.22. Success Status

cmi.success_status データモデル要素は，学習者が SCO を終了したかどうかを示す[1]．SCO がどのように cmi.success_status を決定するかは，SCORM の対象範囲外である．SCO はこの決定をいくつかの方法に基づくことできる：合格した相互作用の比率，達成された目標の比率，テストやクイズに対する合計スコアとマスタースコアの比較等々．この値は，SCO 開発者に決定されるように，SCO に対する全体的な成功ステータスを示す．

表 4.2.22a: Success Status データモデル要素に対する Dot-notation バインディング t

Dot-Notation Binding	Details
cmi.success_status	**<u>データモデル要素実装要件</u>:** • **データタイプ**: 状態 (passed, failed, unknown) • **値空間**: IEEE ドラフトは 3 つの状態値を定義する．SCORM はこれらの値を以下の制限された語彙トークンにバインドする： o "passed": 学習者が SCO をパスした[1]．必要な数の目標がマスターされたもしくは必要なスコアが達成されたことを示す o "failed": 学習者が SCO を失敗した[1]．学習者が必要な数の目標をマスターしなかったもしくは要求されたスコアが達成されなかったことを示す o "unknown": 主張ができない[1]．成功ステータスを示す主張が適用できないことを示す • **フォーマット**: このデータモデル要素のフォーマットは上記の 3 つの語彙トークンのうちの一つ ("passed", "failed", "unknown") *cmi.objectives.n.success_status* データモデル要素は学習者が目標をマスターしたかどうかを示す[1]．SCO がどのように目標に対して *cmi.objectives.n.success_status* を決定するかは SCORM の対象範囲外である．SCO はいくつかの根拠によりこの決定を下すことができる：目標にマッピングする対話の比率，テストやクイズの合計スコア，目標に基いて，マスタースコアに対比される等々．この値は，SCO 開発者に決定されたように，SCO に対する overall success status を示す． 目標ステータスに依存し，SCO と関連する学習アクティビティに適用されたシーケンシング情報があれば，SCO は，SCO の学習者セッション終了の前に目標情報が正確に LMS に送られたことを確実にしなければならない．そうでないと，シーケンシング情報を処理するとき，LMS は，SCO と関連する学習アクティビティの目標に対する目標ステータスとして"unknown"値を利用する **<u>LMS 動作要件:</u>** • このデータモデル要素は必須で LMS は read/write として実装しなくてはならない • *cmi.success_status* の決定は最初 SCO にコントロールされ管理されるので，LMS は *cmi.success_status* の値を示唆することができない．SCORM には SCO が *cmi.success_status* を設定することを義務付ける要件はない．SCO が *cmi.success_status* を設定しないと，LMS は *cmi.success_status* の値としてデフォルト値の"unknown"を利用する．しかし，*cmi.scaled_passing_score* が定義され，*cmi.scaled.score* が SCO にレポートされると，LMS はセクション 4.2.22.1: *Success Status Determination* で定義された要件に準拠して *cmi.success_status* を上書きする **<u>シーケンシングインパクト:</u>**

	cmi.objectives.n.success_status および *cmi.objectives.n.score.scaled* からのシーケンシングインパクトが考慮された後に以下が発生する • SCO もしくは LMS が SCO の *cmi.success_status* を“unknown”に設定すると，SCO に関連する学習アクティビティの主要目標に対する *Objective Progress Status* は false である． • SCO もしくは LMS が SCO の *cmi.success_status* を“passed”に設定すると，SCO と関連する学習アクティビティの主要目標に対する *Objective Progress Status* は true になり，SCO と関連する学習アクティビティの主要目標に対する *Objective Satisfied Status* は true となる • SCO もしくは LMS が SCO の *cmi.success_status* を“failed”に設定すると，SCO と関連する学習アクティビティの主要目標に対する *Objective Progress Status* は true になり，SCO と関連する学習アクティビティの主要目標に対する *Objective Satisfied Status* は false となる **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.success_status* データモデル要素の値を検索および保存することが許されている． • SCO は，LMS に遵守される *cmi.success_status* 決定ルールに留意すべきである．これらのルールは，どのようにおよびいつ LMS が SCO にレポートされる *cmi.success_status* を上書きするか記述する • SCO は， should be aware that setting the *cmi.success_status* を設定すると SCO と関連する学習アクティビティに影響があり，従って恐らくシーケンシングに影響があることに留意すべきである • 成功ステータスに依存し，SCO と関連する学習アクティビティに適用されたシーケンシング情報があれば，SCO は SCO の学習者セッション終了の前に成功情報が正確に LMS に送られることを確実にしなければならない．そうでないと，LMS は，シーケンシング情報を処理するとき，SCO と関連する学習アクティビティの主要目標に対する目標ステータスとして “unknown”値を利用する **API 実装要件:** • **GetValue():** LMS は，学習者に対して現在 LMS に保存された関連する *cmi.success_status* を返し，エラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する． o 何らかの決定要因が存在するまで，*cmi.success_status* のデフォルト値は“unknown”である • **SetValue():** SCO が *cmi.success_status* を設定する要求を呼び出し，値が上記の制限された語彙トークンのうちの一つでなければ，LMS はエラーコードを“406” – Data Model Element Type Mismatch に設定し“false”を返す．LMS はこの要求にもとづていデータモデル要素の状態を変える事はない **例:** • GetValue(“cmi.success_status”) • SetValue(“cmi.success_status”,”passed”)

4.2.22.1 Success Status 決定

一般的に，SCO の成功ステータスは，SCO によって決定される．SCO の成功ステータスは，コンテンツ開発者に様々な方法で設計される：

- **質問に対してあるパフォーマンス測定値を入手する**
- **目標もしくは SCO と関連する一組の目標に対してあるパフォーマンス測定値を入手する**

SCO によって，どのように決定されるかに関わらず，このプロセスは，SCO が cmi.success_status を設定することに関与する．SCORM は，SCO に SCORM ランタイム環境データモデル要素を追跡することを要求しない．これを念頭に，LMS は，成功ステータスの決定に要求される追加動作を持つことがある．状況によって LMS は異なった動作をするように要求される．

cmi.success_status データモデル要素は，他の二つの SCORM ランタイム環境データモデル要素に影響される(スケールされた合格スコア – cmi.scaled_passing_score およびスケールされたスコア – cmi.score.scaled)．以下の表は，これらの値と，定義された LMS 動作の状態を定義する．スケールされた合格スコアは，imsmanifest.xml ファイルで定義される(セクション 4.2.19: *調整された合格スコア*参照)．調整されたスコアおよび成功ステータスは，両方とも SCO によって決定される．表 4.2.22.1a は，これらの値の設定され定義される，あるいは，設定されない定義されないという組み合わせに関連付けられた LMS 動作を定義する.LMS 動作は，cmi.success_status の値を取得する GetValue()要求が SCO によって発せられた時に適用される．LMS に格納された cmi.success_status の実際の値が上書きされたり決定の過程で持続するかどうか，は SCORM 規格の範囲外であり実装依存である．

表 4.2.22.1a: Success Status 決定

スケールされた合格スコア	スケールされたスコア	成功ステータス	LMS 動作
定義なし	SCO に設定された値なし	SCO に設定された値なし	cmi.success_status が“ unknown ”に設定され，この値は SCO に返される
定義なし	SCO に設定された値なし	定義された語彙の一つ	アクションなし．LMS に現在管理される cmi.success_status(すなわち，SCO からの最後に成功した SetValue()呼び出しで設定された値) は SCO に返される
定義なし	0.5	定義された語彙の一つ	アクションなし．LMS に現在管理される cmi.success_status(すなわち，SCO からの最後に成功した SetValue()呼び出しで設定された値) は SCO に返される
0.8	0.5	定義された語彙の一つ	cmi.success_status は failed に決定され，SCO に返される 理論的解釈: 0.5 < 0.8． cmi.scaled_passing_score および cmi.score.scaled で定義された要件参照
0.8	0.9	定義された語彙の一つ	cmi.success_status は passed に決定され，SCO に返される 論理的解釈: 0.9 >= 0.8． cmi.scaled_passing_score および cmi.score.scaled で定義された要件参照
0.8	SCO に設定された値なし	SCO に設定された値なし	cmi.success_status は unknown に決定され，SCO に返される
0.8	0.5	SCO に設定された値なし	cmi.success_status は failed に決定され，SCO に返される 論理的解釈: 0.5 < 0.8. cmi.scaled_passing_score および cmi.score.scaled で定義された要件参照

0.8	0.9	SCO に設定された値なし	cmi.success_status は passed に決定され, SCO に返される 論理的解釈: 0.9 >= 0.8. cmi.scaled_passing_score および cmi.score.scaled で定義された要件参照
定義なし	0.5	SCO に設定された値なし	cmi.success_status は unknown に決定され, SCO に返される
0.8	SCO に設定された値なし	定義された語彙の一つ	cmi.success_status は unknown に決定され, SCO に返される

ADL Note: Scaled Passing Score は SCO リソースが参照する<imscp:item>要素の <imsss:primaryObjective>要素に関連する IMS シンプルシーケンシング名前空間属性 imsss:satisfiedByMeasure によって定められる. imsss:satisfiedByMeasure 属性が true に設定されるなら, <imsss:minNormalizedMeasure>要素で供給される値は LMS によって cmi.scaled_passing_score の初期化に使われる. <imsss:minNormalizedMeasure>要素が定義されないなら, デフォルト値の 1.0 が使われる. imsss:satisfiedByMeasure 属性が false に設定されるなら, 定義された minimum normalized measure は存在せず, cmi.scaled_passing_score は LMS による初期化は行われない.

4.2.23. Suspend Data

学習中，学習者もしくは SCO は，SCO への学習者の試行を中断し，後に学習者試行を再開したいと思うことがある．SCO への学習者試行が中断されると，SCO のランタイムデータの状態は，SCO への次の学習者セッションまで保持される (cmi.exit が “suspend”に設定されると)．cmi.suspend_data データモデル要素は，学習者セッション間の中断データを保管および検索する追加スペースを提供する．中断データは，学習者試行を再開するために，SCO によって利用される．

cmi.suspend_data データモデル要素は，学習者アクセスもしくは SCO との相互作用の結果として SCO によって作成されるかもしれない情報を提供する．このデータモデル要素のコンテンツのフォーマットは，特定されていない[1]．

目的は，SCO が，その SCO および同一の学習者間の現在もしくは後の学習者セッションで，後に利用するためにデータを保存することである．LMS は，このデータを解釈も変更もしない[1]．

このデータモデル要素は，一般的に cmi.location が利用できない中断状態から SCO が再開に必要とする情報を保存するのに利用される．データは，同一の学習者セッションで後に利用されることが可能である．

表 4.2.23a: Suspend Data データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.suspend_data	**データモデル要素実装要件:** • **データタイプ:** 文字列(SPM: 4000) • **値空間:** ISO-10646-1 • **フォーマット:** この文字列のフォーマットは SCO 開発者次第である．LMS は単に SCO の要求に基づいてデータを保管し返すだけである．詳細はセクション 4.1.1.7: *データタイプ*参照 **LMS 動作要件:** • このデータモデル要素は必須で LMS は read/write.として実装しなくてはならない • LMS はこのデータの初期化に責任がない．LMS は SCO に対してデータの保管と検索だけに責任がある．SCO が *cmi.suspend_data* が設定される前に値を要求すると，LMS は下記の API 実装要件に従って動作する **SCO 動作要件:** • LMS はこのデータモデル要素を read/write として実装する必要がある．SCO は *cmi.suspend_data* データモデル要素の値を検索および保存することが許されている **API 実装要件:** • **GetValue():** LMS は，学習者に対して現在 LMS に保存された関連する *cmi.suspend_data* を返し，エラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する． o SCO が要求を得る前に値を設定しないと LMS はエラーコードを“403” - Data Model Element Value Not Initialized に設定し空の文字列(“”)を返す • **SetValue():** LMS は *cmi.suspend_data* を提供されたデータに設定し，エラーコードを“0” - No error に設定し，“true”を返す．この値はデータモデル要素実装要件で定義された有効な文字列でなければならない **追加動作要件:**

	• SCO は LMS に保存された情報に責任がある．LMS に送られたデータは，どんな文字列の提示でも有効であれば許されている．LMS は最低 SPM の 4000 キャラクターを提供する．4000 キャラクター以上の文字列は全部が LMS に保存されることを保証されていない．目的は，SCO が現在のもしくは後の学習者セッションで後の利用のためにデータを保存することである．SCO は *cmi.suspend_data* のフォーマットに対して責任があるので，LMS はこのデータを解釈もしくは変更すべきでない • LMS は，前回の学習者セッションで SCO が *cmi.exit* の値を “suspend”に設定した場合に限り，このデータを前回の学習セッションで設定されたように提供する．言い換えれば，LMS は，*cmi.suspend_data* を，同一の学習者セッション中は，*cmi.exit* の値に関わらず，利用可能とする．しかし，SCO が，*cmi.exit* を“suspend”に設定しないで Terminate(“”)を要求すると，LMS は *cmi.suspend_data* の値を破棄し，後のセッションでこのデータを SCO に戻さない．この動作は前のセッションで設定されたどんな他のデータにも適用する **例:** • SetValue(“cmi.suspend_data”,”<data><intID>1001</intID><ans>A</ans></data>”) • SetValue(“cmi.suspend_data”,”A1;B2;C11-3”)

4.2.24. Time Limit Action

cmi.time_limit_action データモデル要素は，cmi.max_time_allowed が超えたときに SCO が何をすべきか示す [1].

表 4.2.24a: Time Limit Action データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	説明
cmi.time_limit_action	**データモデル要素実装要件:** • **データタイプ:** 状態 (exit_message, continue_message, exit_no_message, continue_no_message) • **値空間:** IEEE ドラフトは 4 つの状態値を定義する．SCORM はこれらの状態値を以下の制限された語彙トークンにバインドする： o “exit,message”: 学習者は SCO をエグジットさせられる．SCO は学習者に学習者アテンプトに与えられた時間が超過したことを示すメッセージを提供すべきである[1] o “continue,message”: 学習者は SCO を継続することが許される．SCO は学習者に学習者アテンプトに与えられた時間が超過したことを示すメッセージを提供すべきである[1] o “exit,no message”: 学習者はメッセージなしで SCO をエグジットさせられる[1] o “continue,no message”: 学習者は学習者アテンプトに与えられた時間を超過したが，メッセージは与えられず，SCO をエグジットさせられることもない[1]．これが，このデータモデル要素のデフォルト値である **LMS 動作要件:** • このデータモデル要素は必須で LMS は read-only として実装しなくてはならない • LMS は，ADL Content Packaging namespace 要素<adlcp:timeLimitAction>を利用して，このデータモデル要素を初期化する責任がある．この要素は，コンテンツパッケージマニフェストにあり SCO リソースを参照する <imscp:item>要素にだけおかれる．If the <adlcp:timeLimitAction>が<imscp:item>に定義されていないと，LMS はこの値を“continue,no message”（デフォルト値） に初期化する **SCO 動作要件:** • LMS はこのデータモデル要素を read-only として実装する必要がある．SCO は *cmi.time_limit_action* データモデル要素の値を検索することだけが許される **API 実装要件:** • **GetValue():** LMS は *cmi.time_limit_action* データモデル要素に対して保存された値を返しエラーコードを“0” – No error に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する． o マニフェストに定義された時間制限アクションがなく，SCO がこのデータモデル要素の値を要

	求すると，LMS は“continue,no message” (デフォルト値) を返しエラーコードを“0” – No Error に設定する • **SetValue():** SCO が *cmi.time_limit_action* を設定するのに SetValue()要求を呼び出すと，LMS はエラーコードを “404” – Data Model Element Is Read Only に設定し “false”を返す．LMS はこの要求に基づいてデータモデル要素の状態を変えることはない **例:** • GetValue(“cmi.time_limit_action”)

4.2.25. Total Time

cmi.total_time データモデル要素の値は，現在の学習者セッションの前に現在の学習者試行内で累計された学習者の全セッション時間(cmi.session_time)の合計である[1]．このデータモデル要素は，ある学習者試行に対して，全ての学習者セッションで費やされる時間の合計を追跡するのに利用される（セクション 2.1.1: *ランタイム環境時間モデル*参照）.

表 4.2.25a: Total Time データモデル要素に対する Dot-notation バインディング

Dot-Notation バインディング	**説明**
cmi.total_time	**<u>データモデル要素実装要件:</u>** • **データタイプ:** timeinterval (second,10,2) – **精度 0.01 秒の時間インターバル** • **フォーマット:** 詳細はセクション 4.1.1.7: *データタイプ*参照 **<u>LMS 動作要件:</u>** • **このデータモデル要素は必須で** LMS **は** read-only **として実装しなくてはならない** • **このデータモデル要素は** LMS **に** read-only **として実装されるので，このデータを管理するのは** LMS **の責任である．この値は累計されたセッションタイムなので(*cmi.session_time*)，**LMS **は，**SCO **がセッションタイムを設定するまでこの値を決定できない．**SCO **が，セッションタイムが設定される前に，値を要求すると，**LMS **は以下の** API **実装要件に従って動作する** • *cmi.total_time* **の値は，学習者セッションが進行中はアップデートされない** • *cmi.total_time* **のデフォルト値は** PT0H0M0S **である** **<u>SCO 動作要件:</u>** • LMS **はこのデータモデル要素を** read-only **として実装する必要がある．**SCO **は，*cmi.total_time* データモデル要素の値を検索することだけが許されている** **<u>API 実装要件:</u>** • **GetValue():** LMS **は学習者に対して現在** LMS **に保存された関連する *cmi.total_time* を返しエラーコードを**"0" – No error **に設定する．返された文字列はデータモデル要素実装要件で特定された要件を遵守する．** o SCO **がセッションタイムを設定していないと，**LMS **は *cmi.total_time* 値 を決定できない．これらのケースでは，**LMS **はデフォルト値の** PT0H0M0S **を返しエラーコードを** "0" – No Error **に設定する** • **SetValue():** SCO **が *cmi.total_time* を設定するのに要求を呼び出すと，**LMS **はエラーコードを** "404" – Data Model Element Is Read Only **に設定し** "false"**を返す．**LMS **はこの要求に基づいてデータモデル要素の状態を変えることはない** **<u>追加動作要件</u>**: • SCO **は，一回の交信セッションで，複数のセッションタイム (*cmi.session_time*)を設定することが許されている．**SCO **が** Terminate("")**を出すともしくはユーザーが離れて行くと，**LMS **は最後の *cmi.session_time* を取得し，この時間を *cmi.total_time* に累計する** **<u>例:</u>** • GetValue("cmi.total_time")

付録 A
略語表

このページは空白である.

略語表

ADL	Advanced Distributed Learning
AICC	Aviation Industry CBT Committee
API	Application Programming Interface
ARIADNE	Alliance of Remote Instructional Authoring & Distribution Networks for Europe
CAM	Content Aggregation Model
CMI	Computer Managed Instructions
DOM	Document Object Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
LMS	Learning Management System
LTSC	Learning Technology Standards Committee
RTE	Run-Time Environment
RTS	Run-Time Service
SCO	Sharable Content Object
SCORM	Sharable Content Object Reference Model
SN	Sequencing and Navigation
SPM	Smallest Permitted Maximum
SS	Simple Sequencing
UI	User Interface
URI	Universal Resource Identifier
URL	Universal Resource Locator
URN	Universal Resource Name
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language

このページは空白である.

付録 B
参考文献

このページは空白である．

参考文献

1. *IEEE 1484.11.1 Standard for Learning Technology – Data Model for Content Object Communication*. January 14, 2005.
   Available at: http://www.ieee.org/
2. *IEEE 1484.11.2-2003 Standard for Learning Technology – ECMAScript Application Programming Interface for Content to Runtime Services Communication*. November 10,2003
   Available at: http://www.ieee.org/
3. *IETF RFC 2141*: 1997, URN Syntax.
   Available at: http://www.ietf.org/
4. *ISO/IEC 646:1991, Information technology – ISO 7-bit coded character set for information interchange*.
5. *ISO/IEC 10646-1, Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane.*
6. *IETF RFC 3986:2005, Uniform Resource Identifier (URI): Generic Syntax*.
   Available at: http://www.ietf.org/
7. *Aviation Industry CBT Committee (AICC) Computer Managed Instruction (CMI) Guidelines for Interoperability Version 3.5*. April 2, 2001
   Available at: http://www.aicc.org/
8. *W3C, Document Object Model (DOM) Level 3 Core Specification, Version 1.0,* W3C Working Draft 26 February 2003. (See: http://www.w3.org/DOM/)
9. *ISO/IEC 16262:1998, Information technology—ECMAScript language specification*
10. *SCORM 2004 3rd Edition Overview Version 1.0,* Advanced Distributed Learning, October 20, 2006
    Available at: http://www.adlnet.gov/
11. *SCORM 2004 3rd Edition Sequencing and Navigation Version 1.0,* Advanced Distributed Learning, October 20, 2006
    Available at: http://www.adlnet.gov/
12. *ISO/IEC 11404:1996, Information technology – Programming languages, their environments and system software interfaces – Language-independent datatypes*
13. The Unicode Consortium. *The Unicode Standard, Version 4.0.0*, defined by: *The Unicode Standard, Version 4.0* (Boston, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1)
14. *ISO 639–1, Code for the representation of names of languages – Part 1: Alpha-2 code*.

15. *ISO 639–2, Codes for the representation of names of languages – Part 2: Alpha-3 code*.

16. *ISO 3166–1, Codes for the representation of names of countries and their subdivisions – Part 1: Country codes.*

17. *IMS Simple Sequencing Behavior and Information Model v1.0 Final Specification*, IMS Global Learning Consortium, Inc., March 2003
Available at: http://www.imsproject.org/.

18. *SCORM 2004 3rd Edition Content Aggregation Model Version 1.0,* Advanced Distributed Learning, October 20, 2006
Available at: http://www.adlnet.gov/

付録 C
ドキュメント改定履歴

このページは空白である.

ドキュメント改定履歴

SCORM バージョン	リリース日付	変更詳細
1.3 ワーキングドラフト 1	2003 年 10 月 20 日	初期ドラフト • IEEE P1484.11.1 Draft 1/WD 13 Draft Standard for Learning Technology – Data Model for Content Object Communication による変更 • IEEE P1484.11.2 Draft 4 Standard for Learning Technology – ECMAScript Application Programming Interface for Content to Runtime Services Communication による変更 • IMS Simple Sequencing Version 1.0 を反映する変更 • IMS Content Packaging Version 1.1.3 を反映する変更
RTE バージョン 1.3	2004 年 1 月 30 日	以下に基づいたアップデート： • IEEE P1484.11.1, Draft 3 Draft Standard for Learning Technology – Data Model for Content Object Communication による変更 • IEEE 1484.11.2 – Standard for Learning Technology – ECMAScript Application Programming Interface for Content to Runtime Services Communication による変更
cmi.interactions.n.objectives.m.id	2004 年 7 月 22 日	• 様々な文法上の修正，印刷上の修正およびスタイルの一貫性 • パラメータが空の文字列の GetDiagnostic() API メソッドの処置に対する推奨の追加 • API メソッドがどの状態の間に要求されるのが許されるかの記述を追加するために図 3.1.6a をアップデート • SPM が超過されたときの推奨される LMS 動作を記述するセクション 3.1.7.6.7 Smallest Permitted Maximum Exceeded エラー条件の追加 • _version keyword が不正に他の SCORM ランタイム環境データモデルと利用さ

		れたときに推奨される LMS 動作を記述するセクション 3.1.7.6.8 Data Model Element Does Not Have Version エラー条件を追加 • API インスタンスがおかれる正しい場所の記述を強化するためにセクション 3.2.1 API Instance をアップデート．3 つの異なったロケーションがあることを図示するイメージを強化するために図 3.2.1a をアップデート． • API バージョン要求を記述するためにセクション 3.2.1.1 バージョン属性を追加 • SCO が探索し API インスタンスを見つけるロケーションを図示するイメージを強化するために図 3.3.1a をアップデート • 図 3.3.1.b をアップデート．図をコード図解に変更しアルゴリズムを記述するテキストを追加 • 図 4.1.1.6a Reserved Delimiter にあるべきなのになかったデリミター[:]を追加 • セクション 4.1.1.7 データタイプをアップデート：<language_type> indicator for the localized_string_type に対してあるべきなのになかった SPM を追加 • IEEE Data Model for Content Object Communication の改定に基づいて language_type をアップデート • あるべきなのになかった SPM の 4000 キャラクターを long_identifier_type に追加 • あるべきなのになかった SPM の 250 キャラクターを short_identifier_type に追加 • IEEE Data Model for Content Object Communication の改定に基づいて time(second,10,2)を time(second,10,0)にアップデート • ISO 8601 に制約される time(second,10,0)文字列により多くの要件を追加した．特に Time Zone Indicator の処置に • cmi.comments_from_learner をアップデート． IEEE Data Model for Content Object Communication の変更を反映するために親データモデル要素 date_time を timestamp に変更 • cmi.comments_from_lms をアップデー

		ト．IEEE Data Model for Content Object Communication の変更を反映するために子データモデル要素 date_time を timestamp に変更 • cmi.exit 値の“logout”が cmi.entry に LMS によって“resume”に設定されることを起こすことを示すために cmi.entry (LMS 動作要件)をアップデート．この値はあるべきなのになかった • cmi.interactions.n.id, cmi.objectives.n.id および cmi.interactions.n.objectives.m.id データモデル要素に対する無効な GetValue()コールの処置を削除．識別子が設定される前に識別子を取得するために GetValue()要求がなされた場合の要求を処置する動作 • cmi.interactions.n.timestamp データタイプを time(second,10,2)から time(second,10,0)へアップデート • セクション 4.2.9.1 のタイトルをセクションをより良く表すために Correct Responses Pattern Data Model Element Specifics に変更 • IEEE Data Model for Content Object Communication の改定に基づきセクション 4.2.9.1 をアップデート： o 文字列値の要件とフォーマットの理解を容易にするための強化 o choice correct response を a set of set of short_identifier_values. にアップデート．これは a bag of set of short_identifier_values であった．これらの変更は IEEE Data Model for Content Object Communication の改定に基づいている．ユニークなセットおよび各セットのユニークな short_identifier_values の追加要件を特定した．short_identifier_values の順序は重要でないことを定義した o choice correct response pattern がゼロ以上の choice correct responses を許すようにアップデートした．correct responses がないことが示されれば，対話に correct response がないことを示唆する o IEEE Data Model Content Object

		Communication の改定に基づいて各対話タイプの要件を明確化した • セクション 4.2.9.2 Learner Response Data Model Element Specifics をアップデートした： o choice interaction type の記述を a bag of short_identifier_type から a set of short_identifier_type に変更. Update requirements to reflect the change to a set への変更を反映する要件をアップデート (文字列のユニークば short_identifier_type 値) • 誤った値の“wrong”を修正するのに cmi.interactions.n.result をアップデート. “wrong”を“incorrect”に変更 • cmi.launch_data で誤った参照 <adlcp:datafromlms>を <adlcp:dataFromLMS>へアップデート • cmi.learner_preference.audio_level に対する Value Space を値は 0 以上であることを記述するためにアップデート • cmi.learner_preference.audio_level に対するデフォルト値が 1 である定義を追加 • cmi.objectives.n.progress_measure に対する情報を追加. これは IEEE Data Model for Content Object Communication に紹介された新たな要素である. • cmi.time_limit_action データモデル要素で誤った参照<adlcp:timelimitaction>を <adlcp:timeLimitActino>にアップデート • cmi.time_limit_action に対するデフォルト値(“continue,no message”)の定義を追加
SCORM 2004 3rd Edition: RTE Version 1.0	2006 年 10 月 20 日	以下に基づいたアップデート： • ドキュメント中の全ての画像を更新. 色とイメージの外観をより一般的なものに変更. • meta-data を medadata に変更. ハイフンを除去. • セクション 2.1.1.2 Persisting Run-Time Data Across Learner Attempt and Activities を除去. • SCORM テクニカルワーキンググループ会議により, セクション 4.1.1.2 Data Model Effects on Sequencing を更新.

		Abandon または Abandon All ナビゲーション要求が起こった時のランタイム環境データの管理に関する用語を含む. • セクション 4.1.1.3 Handling Collections を更新, コレクションの新レコードの作成の扱いに関する変更を反映. • セクション 4.1.1.6 Reserved Delimiters を更新. デリミタをデリミタのタイプ(予約された属性デリミタと予約された分割デリミタ)に分割. • セクション 4.1.1.7 Data Types を更新. Unicode と SPM サポートに関して, 文字列データ型とその利用についての説明を付加. • セクション 4.1.1.7 Data Types を更新. langcode と subcode の値の長さについて述べる文言を追加. • セクション 4.1.1.7 Data Types を更新. long_identifier_type と short_identifier_type が空文字列や空白文字列にならないデータ型である, という定義を追加. • セクション 4.1.1.7 Data Types を更新. time(second,10,0)の YYYY の値の範囲についての誤りを訂正. 変更は行われることになっていた(1970 <= YYYY <= 2038). デフォルトのタイムゾーン指定子が何になるか, の説明を更新. • セクション 4.1.1.7 Data Types を更新. timeinterval (second, 10,2)がゼロ埋めの値をサポートするように定義. • completion_status の決定時における LMS 動作要求を更新. この動作は cmi.completion_threshold が定義されたか, completion_status がいつ決定されたかに基づいて行われる. この変更にともない, API Implementation Requirements も更新した. • cmi.completion_status が"not attempted" に設定された時の cmi.completion_status データモデル要素の Sequencing Impacts を更新. • 表 4.2.4.1a: Completion Status Determination を更新. > を >= に. • セクション 4.2.7 Entry を更新. LMS が Suspend All ナビゲーション要求を発した時に"resume"の値も使われることを

		示した. • セクション 4.2.8 Exit を更新. 学習アテンプト間のランタイム環境データモデル要素の扱いを明示. • セクション 4.2.8 Exit を更新. "logout"の値への批判があり実装では使われるべきでない, という注を追加. • セクション 4.2.9 Interactions (cmi.interactions.n.objectives.m.id) を更新. 使われる値が配列上の他の位置で既に LMS によって持たれている値と同一である時に LMS が SetValue()呼び出しをどのように扱うか, の説明を追加. • cmi.learner_preference.language のデータモデル要素実装要件を更新. データ型が空文字列をサポートするように. • セクション 4.2.17.2 Initialization of Run-Time Objectives from Sequencing Information を更新. • cmi.objectives._children から返される値のリスト progress_measure の子要素を追加. • cmi.objectives.n.id (API Implementation Requirements)を更新. LMS 動作において, 「配列上のより早い位置で既に設定された識別子と同一」の識別子の扱いを定義. (すなわち, SetValue()要求を処理する時の重複した識別子の扱い) • セクション 4.2.19 Scaled Passing Score を更新. LMS の初期化動作について記述. • セクション 4.2.21 Session Time を更新. セッション時間の初期化とそれがいつ発生するか, について説明を追加. • success_status の決定に関する LMS 動作要件を更新. この動作は cmi.scaled_passing_score が定義されたか, success_status がいつ決定されたかに基づいて行われる. この変更にともない, API Implementation Requirements も更新した. • 表 4.2.22.1a Success Status Determination を更新. • cmi.suspend_data の SPM 値を更新. 4000 文字から 64000 文字に.

		• セクション 4.2.25 Total Time を更新. デフォルト値は 0 の期間に等しい. • ドキュメントのバージョンを SCORM 2004 3rd Edition になるように更新. 内部バージョンを 1.0 に更新. • ドキュメント全体の必要箇所において, SCORM 2004 3rd Edition への変更に関する更新. • ADL Co-Lab Hub のコンタクト情報を更新 • www.adlnet.org への参照を www.adlnet.gov に変更. • ドキュメント全体における編集上の変更(例えば, 一貫して SCORM 登録商標が適用される). • SCORM ドキュメント中の変更を反映する一般的な更新(すなわち, Content Aggregation Model と Sequencing and Navigation books). • SCORM Extension Error Condition を追加. Identifier Value Can Only Be Set Once のエラー状態は識別子が複数回, 異なる値に設定された場合を記述するために追加した(cmi.objectives.n.identifier). • IEEE Standard for Learning Technology - Data Model for Content Object Communication において"Draft"を削除. この標準は既にドラフト版ではない. • 明確さのために, 2 つの配列インデックスを持つ全てのデータモデル要素の中の"n"を"m"に変更した. 例えば, cmi.interactions.n.objectives.n.id は cmi.interactions.n.objectives.m.id に変更. • time(second,10,0)の下に黒点を追加. TZD が定義された時に(全てがゼロであるかもしれないが)hh:mm:ss.s もまた定義される, ということを示すため. これは IEEE 標準で定められた normative expression のバグによるものである. • timeinterval(second,10,2)を更新. 模範的な表現と周囲の用語によって定義されるシンタックスを明確にするため. この変化は, 使う時間構成要素がないならば厳密な指定子 T を使わない, ということに対応した. • cmi.exit 要素を更新. その値が"suspend"に設定された時の動作を明確にするた

		め. 現在のモデルに対するインパクトとデータ持続への影響に関する記述は, 明快さのために加えられた. • cmi.exit データモデル要素に対する注を追加. "logout"キーワードが批判の対象であることを説明. • cmi.interactions.n.type データモデル要素の欠けていたエラー状態を追加. SetValue() API 実行要件において, エラー状態 408 - Data Model Dependency not Established を追加. • cmi.max_time_allowed データモデル要素を更新. LMS 動作要件はこのデータモデル要素が attemptAbsoluteDurationLimit 要素と関連していたことを示す. attemptAbsoluteDurationLimit は limitConditions 要素の属性である. • cmi.objectives データモデル要素の子要素の集合を説明する際に使われていた "collection"という用語を削除. • 以降の学習セッションでの cmi.objectives データモデル要素を初期化するための規範的な用語を更新した. LMS は関連する cmi.objectives データモデル要素をトラッキングモデルから利用可能な最も直近の情報で更新するように要求される. • cmi.objectives.n.id データモデル要素への新しい動作要件を追加. データモデル要素が値を持ったら, SCO がそれを書き換えることは許されない. LMS が従うエラー状態と動作も追加した. • cmi.objectives.n.score.scaled に対して定められた用語を更新. 規範的な用語の使用は正確でなかった. 規範的な用語はより有益な用語に変更した. SCO は値を設定することを全く求められていない. その用語は SCO が値を設定しなくてはならないことを示していた. • データモデルの値が確立されていない場面での cmi.objectives.n.success_status への GetValue()要求の動作が不足していたため, 追加を行った.この場合はデフォルト値の"unknown"が使われる. • データモデルの値が確立されていない場面での

		cmi.objectives.n.completion_status への GetValue()要求の動作が不足していたため, 追加を行った.この場合はデフォルト値の"unknown"が使われる. • LMS が管理する配列中にない学習目標を要求する場面での cmi.objectives.n.progress_measure への GetValue()要求の動作が不足していたため, 追加を行った. この場合は LMS は空文字列を返し, エラーコードを 301 に設定する. 一貫性を保つため, 同じ条件下での SetValue()に対応する動作も追加した. この場合は LMS は false を返し, エラーコードを 351 に設定する. • 不足していたエラー状態 408 - Data Model Dependency Not Established を cmi.objectives.n.progress_measure 用に追加した. • cmi.score.scaled における用語を更新した. 規範的な用語は正確さを欠いていた. 規範的な用語は有益な用語に変更した. SCO は値を設定することを全く求められていない. その用語は SCO が値を設定しなくてはならないことを示していた. • Suspend Data データモデル要素における用語を更新した. 中断されていた学習アテンプトを扱う用語を Exit データモデル要素に変更した. • 中断データの SPM を 4000 から 64000 に変更した. 追加動作要件のセクションにおいて SPM は正しくなく, 先に定義された値である 64000 と合致していなかった. • IETF RFC 2396 への参照を IETF RFC 3986(新版)に変更した. • セクション 4.2.17.2 Initialization of Run-Time Objectives from Sequencing Information への用語を追加した. 以降の学習セッションにおいて, cmi.objectives データモデル要素の値は LMS のシーケンシングの実行によってトラッキングされる学習目標の値に設定される.