Dev Walkthrough for OneApp Learn Screens and Content Structures

Contents

[Design Goals 2](#_Toc236724967)

[Design Ideas for an Authoring System 2](#_Toc236724968)

[Assumptions 3](#_Toc236724969)

[System/Xml Identities 3](#_Toc236724970)

[Packaging 3](#_Toc236724971)

[Image SRC and Final SRC 3](#_Toc236724972)

[Content Changes 4](#_Toc236724973)

[Home Screen and About Page 5](#_Toc236724974)

[ProductDetails 7](#_Toc236724975)

[AssessmentContent 10](#_Toc236724976)

[Calculating Correctness 10](#_Toc236724977)

[Calculating Overall Score 11](#_Toc236724978)

[Score Detail View/Learning Guide Composition 11](#_Toc236724979)

[Inner-Topic Navigation ( Activity to activity ) 12](#_Toc236724980)

[“Back” and other Navigation 13](#_Toc236724981)

[TextAndPictureContent 13](#_Toc236724982)

[Scrolling 13](#_Toc236724983)

[ContentAssets 15](#_Toc236724984)

[Note on ContentAssets and Content Outline 15](#_Toc236724985)

[ContentOutline 17](#_Toc236724986)

[SubjectAreas 18](#_Toc236724987)

[Forms and uses of subjectareas.xml documents 19](#_Toc236724988)

[LocationDetails 20](#_Toc236724989)

[Location Town View 20](#_Toc236724990)

[Location Results View 20](#_Toc236724991)

[Location Detail View 21](#_Toc236724992)

[Rating Locations 21](#_Toc236724993)

[Calculating Displayed Rating 21](#_Toc236724994)

[Location Notification 21](#_Toc236724995)

[Maintaining Location Detail Identifiers 22](#_Toc236724996)

[Forms and uses of LocationDetails.xml documents 22](#_Toc236724997)

[FlashcardGameContent 24](#_Toc236724998)

[AdventureGameContent 24](#_Toc236724999)

[TipContent 25](#_Toc236725000)

# Design Goals

This breakdown of a content model is being developed with the foremost in mind:

* Enable a re-useable learning content model for mobile experiences regarding conceptual content.
* Enable performance for mobile devices and device networks, kojax platform components, and integration and for ease of integration into authoring environments.
* Enable later integration into other authoring systems which work with basic XML structures.

### Design Ideas for an Authoring System

This content model is intend to leveraged within an authoring environment. A limited authoring system ( InfoPath Forms ) is provided at this time, due to time constraints.

An expanded authoring system could either leverage the Microsoft InfoPath form editing behavior directly, or provide its own editing user interface bindings to the XML structures. As well, a more primary set of tasks of the authoring environment would be hide the document management details of the XML files themselves, to provide packaging and possibly publishing features themselves. Many of these functions could also be built directly into the InfoPath environment.

# Assumptions

### System/Xml Identities

This model uses an open XML format for its structures. Documents should have schema validation run ( using an ***MSXML validating parser*** ) on them for compliance to schema before being imported or stored to pre-production or production environments.

Validation: the document samples have a relatively referenced schema configuration and the schemas are in the same folder with the sample xml files. This way, they can be used to create your own template ( stubs ) and will be self-validating in many XML development tool environments ( such as XMetal or Visual Studio, or Word ). Generally this enables the “intellesense” behavior during editing.

**Note**:

One exception to the heavily validation comes from the requirement to segment the content for the best authoring and run-time performance. As such, the design can be considered a set of inter-dependant documents. References to other documents are made within one document and vs. versa.

Identity references made across documents use an attribute form of “*[documenttypename]*idref” ( or in the case of inter topic navigation “previdref, nextidref” and a related “[*documenttypename*]src” attribute at the root of the referencing document. An authoring system will need to maintain any changes made to documents and identities changes between documents.

If a system, such as a service & database, were to decompose these documents further for any reason, they should also consider the maintenance of references and identities as well.

### Packaging

A package to be handed off to a content provider will be in the form of a **zip** file. The basic process would be to add all the files and folders of a product working folder to a zip. Name the resulting zip file something notable, like a version of the product name.

### Image SRC and Final SRC

The final To aid in authoring of images two attributes are used to show the author

### Content Changes

If these documents are decomposed when consumed, the consuming would be responsible for maintaining changes to constituent parts upon update or re-consumption. Otherwise it is recommended the system try to maintain the XML structures as complete system entities for easy create, update, removal.

# Home Screen and About Page



The home screen and corresponding about dialog are presented below. The functionality provided on these screens are:

**Title of the application "Learn"**: This should be brand-able via a configuration file.

**Brand-able Banner Artwork**: Also brand-able via a configuration file.

**Icon Buttons**:

 Magnifying glass takes the users to the All Courses workflow

 Blocks takes the users to the My Course Workflow

 Checkmark takes the users to the Scores Workflow

**Bottom Left Menu Item**: Take the user back to the OneApp main menu.

**Bottom Right Menu**: Accesses the following features

 " All Courses": Starts the All Courses Workflow

 "My Courses": Starts the My Courses Workflow

 "Scores": Starts the Scores Workflow

 "Tell a friend": This send s or opens an SMS session which includes text details or automation to enable the user to share the OneApp Learn application, like a shortcut, or instructions.

 "About": This opens the about OneApp Learn modal dialog, which mainly shows the version number of the application and copyright information.

# ProductDetails

**ProductDetails.xsd, ProductDetails.xml**

****

The functionality provided on these screens are:

**Title of the workflow "All Courses"**: simple title of the workflow.

**Bottom Left Menu Item**: Take the user back to the All Courses Results view.

**Bottom Right Menu**: Accesses the following features

 " Start Course ": Starts the Course and adds the course to their My Courses list.

 " Order DVD ": Provides access to the Order DVD workflow

 " Find Location": Provides access to the Locations Workflow

**Displays Product Detail Content**

 "Product Title": Provides a multi-line 60c field for the Product title

 " Byline ": Provides a 20c field for the company or author byline

 " Tabbed Outline and Details": Provides a tab view for short text versions of the product outline ( syllabus ) and product details such as a description. These two field should allow whitespace formatting including ***line breaks***, and ***whitespace characters***.

There should be 1 product detail document per product.

A product detail defines discoverable information about a product including: a Title, a byline, description, short outline.

The following is the model of **ProductDetails**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] version** | Optional simple float value representing a version of the same document |
| **[attr] lang** | String representation of a language code or name ( agreed upon standard ) |
| **Title** | Required 1-60 character product title |
| **Byline**  | Required 1-20 character line for name of author, or provider |
| **Detail**  | Optional 1-500 character, whitespace preserved text representing details about the product components |
| **Outline**  | Optional 1-500 characters, whitespace preserved text representing the outline of the content |
| **Components**  | Required 1 time. List of components |

The following is the model of **Components**:

|  |  |
| --- | --- |
| Name | Description |
| **OneAppContent** | 0-1 OneAppContent element |
| **DVD** | 0-1 DVD element |
| **Locations** | 0-1 Locations element |
| **AccessPrograms** | 0-1 AccessProgram element |

The following is the model of **OneAppContent**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr]contentoutlinesrc** | (Defaults to **contentoutline.xml** ) URI of product content outline file |
| **[attr]contentassetssrc** | (Defaults to **contentassets.xml**) URI of product content assets file |

The following is the model of **DVD**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr]sku** | Required string representation of a SKU number for ordering  |

The following is the model of **Locations**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr]locationdetailssrc** | (Defaults to **loctiondetails.xml**) String representation of location details source content |
| **(self)** | A list of 1-100 Location elements |

The following is the model of **Location**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr]locationdetailidref** | Identity reference for a location element within a location details source |

The following is the model of **AccessPrograms**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr]offerdetailssrc** | (Defaults to **offerdetails.xml**) String representation of offer details source content |
| **(self)** | A list of 1-100 AccessProgram elements |

The following is the model of **AccessProgram**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr]offerdetailidref** | Identity reference for a program access element within a offer details source |

# AssessmentContent

**AssessmentContent.xsd, AssessmentContent.xml**: A content assessment defines assessment questions linked to topics.



An assessment document contains a groups of questions which are related to specific topics of a course. The groups also define a number of questions to use from within the group as part of the overall assessment.

The scoring application keeps track of the number of correct answers against a given question count for the hosting question group.

Navigation in assessment content should simply allow the following:

* Begin the assessment ( provided through “Continue” menu option )
* Answer the current question
	+ recording the correctness, and advancing to the next question
	+ scoring the assessment, and navigating to a score view
* “Back” or restart the assessment which brings them to the page before the assessment activity

Calculating Correctness

There is only a single question type for assessments, ***multiplechoice, multiplecorrect***. So, for any given question *1 or more correct answers* may be given. For a student to answer the item correctly, they have to select *all the answers attributed as correct*.

If only one answer for a question is attributed as correct, then the behavior is similar to a *multiplechoice, singlecorrect* question. True and false questions can be written this way as well.

Calculating Overall Score

**PassingScore =** value frompassingscore attribute.

**QGs =** all the question groups

**TotalQGCount =** QGs.length

**RunningTotal** = 0

for each question group in **QGs**  {

 **MaxQPG** = total number of questions used from the question group

 **CorrectQPG** = number of correct answers made against the question group

 **TopicScore** = **CorrectQPG / MaxQPG**

 **RunningTotal += TopicScore**

**}**

**StudentScore** = **RunningTotal** / **TotalQGCount**

### Score Detail View/Learning Guide Composition

A score detail view is composed of:

 the score percentage for the test attempt ( determined above )

the pass term “(pass)

( for each topic in the course )

If the score for the topic score above was passing then the topic title goes into a list titled “complete”
else the topic title goes into a list titled “Needs Improvement”

The following is the model of **AssessmentContent**:

|  |  |
| --- | --- |
| Name | Description |
| [attr] version | Optional version number for document ( reserved for tracking purposes ) |
| [attr] contentoutline | URI for the file containing ContentOutline topic identities |
| [attr] passingscore | Optional 0.1-1.0 floating point number for passing score ( default .6 ) |
| Title | Can occur 0-1, 1-60 character string |
| QuestionGroup | Can occur 1-100 times and is a grouping of content items |

The following is the model of **QuestionGroup:**

|  |  |
| --- | --- |
| Name | Description |
| [attr] topicidref | Identity of topic the question group relates to |
| [attr] questioncount | Number of questions to use from the group |
| Question | Can occur 1-100 times |

The following is the model of **Question:**

|  |  |
| --- | --- |
| Name | Description |
| Description | 1-60 character string used to pose the question |
| Choices | Occurs 1 time, contains 1-10 Answer items |

The following is the model of **Answer:**

|  |  |
| --- | --- |
| Name | Description |
| [attr] correct | Default false, required as "true" to indicate a true answer |
| (self) | A 1-40 character string |

# Inner-Topic Navigation ( Activity to activity )

The entry page to a topic is declared in the contentoutline document and this outline can be used for topic to topic navigation. However, on a small device we must break the topic into multiple pages. To enable one activity to link to the other, in the manner of “continuing” the topic. To do this, pages should have a non-ambiguous, “inner-topic” navigation mechanism independent of the contentoutline navigation. In the content, this activity to activity navigation could be authored by providing idref ( **nextidref, previdref** ) attributes at the top of the page.

The OneApp application should enable any of the content pages to navigate to other “non-topic” pages using an inner-content navigation id system.

###  “Back” and other Navigation

In most cases the application logic can define simple mechanism for navigating to the previous screen and define which menu items will be available for the screen as well.

The exception to this is: inner-activity content which provides previdref and nextidref attributes should override the behavior of “Back” or “Continue,” “Next” and so on.

# TextAndPictureContent

This is a content document which provides a few types of text and picture information types including *Introduction*, *Summary*, *Continued*, and *CaptionedImage*.

### Scrolling

TextAndPictureContent Pages would have a single scroll bar which appears upon overflow, the pages should be designed to have no scrolling on medium size and larger devices ( 178x208 ) and minor scrolling on small devices ( 120x160 ).

All **TextAndPictureContent** documents contain the following

|  |  |
| --- | --- |
| Name | Description |
| [attr] version | Optional version number for document ( reserved for tracking purposes ) |
| [attr]contentassetssrc | Optional reference to a contentassets document ( defaults to contentassets.xml )  |
| [attr]previdref | Optional string reference to an content asset within a contentassets document |
| [attr]nextidref | Optional string reference to an content asset within a contentassets document |
| (self) | Exactly one 1 of the following ***Summary, Introduction, Continuation, or CaptionedImage*** |

The following is the model of **Summary**:

|  |  |
| --- | --- |
| Name | Description |
| TopicTitle | Required 1-20 character representation of the topic title ( truncated or abbreviated as needed ) |
| Heading | 1 required 0-20 character subhead, rendered in accent color, small text. |
| Text | 1 required text element containing 1-220 characters of whitespace preserved text. |

Summary is intended to be used to provide a segmented heading for the topic type, to break up the content into sub-sections as needed. It would also serve for term and definition type content in a glossary topic.

The following is the model of **Introduction**:

|  |  |
| --- | --- |
| Name | Description |
| Icon | Require URI for a **20x20** pixel png format icon representing the topic type |
| TopicTitle | Required 1-20 character representation of the topic title ( truncated or abbreviated ) |
| Text | Required 1-220 character whitespace preserved text string |
| ReaderAid | Optional reader aid |

The introduction element is mainly used as a topic landing page, before the main activity, and can be used to quickly identify the type of activity with a small heading and graphic to the user. The remainder of the content on the page provides a terse model for reader preparation purposes using a small text field and a reader aid.

The following is the model of **ReaderAid**:

|  |  |
| --- | --- |
| Name | Description |
| Icon | Require URI for a **30x30** pixel png format icon representing the topic type |
| Text | Required 1-100 character string ( whitespace normalized ) text |

The intent of the reader aid is to provide a short notification to the user about upcoming content, such a preparation materials, or prerequisites.

The following is the model of **Continuation**:

|  |  |
| --- | --- |
| Name | Description |
| TopicTitle | Required 1-20 character representation of the topic title ( truncated or abbreviated as needed ) |
| Text | Required 1-220 character whitespace preserved text string |
| ReaderAid | Optional reader aid |

The intent of the continuation page is to provide a simple text reading experience, without inline formatting. An optional reader aid is allowed at the bottom of the text area.

The following is the model of **CaptionedImage**:

|  |  |
| --- | --- |
| Name | Description |
| TopicTitle | Required 1-20 character representation of the topic title ( truncated or abbreviated as needed ) |
| Caption | Required 1-50 character whitespace normalized text string |
| Image/[attr]src | Photo realistic or rendering 114x114 PNG format image |

# ContentAssets

### Note on ContentAssets and Content Outline

Mobile content communication is somewhat different than the desktop experience for content presentation engines. Many times the content asset information does not need to be retrieved for many of the navigation views. As well, we need to help support a model for delivery of outline data from a content server with restricted bandwidth or network latency. And as need to be aware of client-side cache purge requirements. So, different from IMSManifest, OEB, or other manifest use, content assests ( the definition of the assets used for the content experience ) and content organization are separated here.

Authoring content assets also enables the capability to be SCORM 2001 compliant behavior for content assets and resources management.

**ContentAssets.xsd, ContentAssets.xml**: A content assets document defines the xml, image, ( not yet supported ) file references to be used for individual content experiences.

The following is the model of **ContentAssets**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] version** | Optional simple float value representing a version of the same document |
| **FileGroup** | Can occur 0-999 times and is an arbitrary grouping of files, used to created common file sets as a reference item to be used within a ContentAsset\SupportingFiles section, using IncludeFiles element |
| **ContentItem**  | Occurs 1-9801 ( max number available for an outline). These define content assets for a presentation. |

The following is the model of **FileGroup**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] id** | A unique identity within the document |
| **File** | Occurs 1-999 times, A *FileReference* with corresponding ***src*** attribute. |

The following is the model of **ContentItem**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] id** | A unique identity within the document |
| **InitialFile** | Required 1 time, A *FileReference* with corresponding ***src*** attribute. |
| **SupportingFiles** | Occurs 0-1 times, A group of ***File*** or ***IncludeFiles*** elements. |

The following is the model of **File**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] src** | A common URI form  |

The following is the model of **IncludeFiles**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] idref** | A reference to a unique identity within the document ( a FileGroup element ) |

# ContentOutline

**ContentOutline.xsd, ContentOutline.xml**: A content outline document defines the order of the content topics which would be discoverable for navigation by the content outline view.

The following is the model of **ContentOutline**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] version** | Optional simple float value representing a version of the same document |
| **[attr] contentassets** | (defaults as “**contentassets.xml**”) URI for the file containing ContentAsset identities reference here ( expecting a partial file name here silbling, descendant, or sibling/descendant to the contentoutline.xml document |
| **Items** | Occurs 1 time and is a grouping element for items |

The following is the model of **Items:**

|  |  |
| --- | --- |
| Name | Description |
| **Item** | A navigable item within the content. Can occur 1-99 |

The following is the model of **Item:**

|  |  |
| --- | --- |
| Name | Description |
| **[attr] id** | ID type, represents a unique navigation ID within the content |
| **[attr] contentidref** | String type, representing a unique content asset id (used for upload) |
| **Title** | An 80 character string used for the title of the content item. |

# SubjectAreas

**SubjectAreas.xsd, SubjectAreas.xml**:

The Subject Areas screen provides the following functionality:

**Screen Title**: Short title of the screen contents.

**A Grid View of Icon representing the Subject Area items**: This is a proportional grid with 3 columns and N rows.

**Lower Caption**: For any focued icon, the presents an associated label.

**Bottom Left Menu Item**: Take the user back to the Learn Home Screen.

**Bottom Right Menu**: Provides menu items.

 "Search": Starts the All Courses Workflow



For any Telco provider support region, there should be 1 subject areas document or other format ( master catalog ), which feeds the My Courses feature. For content providers, a single area to product document should be in the package to the service provider.

### Forms and uses of subjectareas.xml documents

The subjectareas.xml document can be authored to contain different first child content. This enables the same schema to be used in different forms to resolve catalogue navigation. For example, the entire catalogue could be composed of XML subjectarea documents which are linked together. For example:

**Possible Physical Structure for Service Provider:**
***For the Subject Area View***
1 document for Subjectarea/Subjects. *See, subjectareassample1.xml.*

***For the Results View***
1 \* number of subjects areas, Subjectarea/ProductResults documents*. See,subjectareassample2.xml .*

**DEV NOTE:** The subjectareas.xml document can also be composed as a single standalone document, if the data is going to be consumed, and indexed by the service… such as when consumed by a SQL design. *( see subjectareassample1.xml )*

The following is the model of **SubjectAreas**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] version** | Optional simple float value representing a version of the same document |
| **Subjects** | 0-1 group of 1-100 Subject elements  |
| **ProductResults** | 0-1 group of 1-1000 Product elements |

The following is the model of **Subject**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] id** | Required unique ID for the subject. |
| **Name** | Required , 40 characters, used for display in the caption area upon control focus |
| **Image** | Required. 1 Image element with src attribute for the subject icon |
| **ProductResultReferences** | Required.(see next) |

The following is the model of **ProductResultReferences**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] subjectareassrc** | Optional URI for source of product results |
| **ProductResultRef** | 0-1000 ProductResultRef elements |

The following is the model of **ProductResultReference**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] internalidref** | Required interal idref for a product result defined within the same document |

The following is the model of **ProductResult**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] productdetailssrc** | Required URI of the product details source document. |
| **ResultTitle** | Required 40 char title, used for display title within the name area of list |
| **ResultContext** | Required 100 char string, used to marquee additional information about the result |
| **ProductResultReferences** | Required. Group  |

**DEV NOTE**: Implied here that if subjectareasrc is not used, the CategoryRef elements within define which categories within the same document are assigned to the subject.

# LocationDetails

**LocationDetails.xsd, LocationDetails.xml**: Location detail defines location data about an iCafe, community center, school, etc.

The Location catalog has a few views which provide different levels of location information:

### Location Town View

This view is accessed from the Learning App spash screen menu. It contains a list of local/provider specific cities which are associated to facility locations.

**DEV NOTE**: The design here assuming different provider areas can be defined across the solution. For example, a learning app can detect or default to different “region” catalogs like having one city catalog for each edge server connection. If not, it might require there be some more logic to accessing different city catalogs for a single provider, such as an DB index system for the catalogs, or more CRUD directly to DB tables.

The term **Region/Town** or **City/County** might be used for different locales.

### Location Results View

This view could use a browselist where provides a collapse/expandable focus item, inside of a typical text based list.

Non-focus items would show the name of the location. Focused items would show, the name, and a “munged” text display of the important, required ( not-all ) location details.

Activating the focus item opens the location detail view for the item

### Location Detail View

This view provides the entire set of location data for a location including name, address, optional description, telephone number, and optional website.

The view also shows the total rating value for the location as well as how many votes were taken.

The menu at this level enables placing a call to the location, and Rating the location.]

### Rating Locations

Rating a location is done by the rating location view which shows a 4-star system list ( less stars for smaller devices ), from Poor to Excellent.

### Calculating Displayed Rating

The starting rating is set to zero stars with a caption in the title of rating as “0 votes” or “no ratings”.

Once votes for a location are submitted, the rating should simply be displayed as:

( product of all the vote values ) / ( the total number of votes for the location )

**DEVNOTE**: The physical design might require tracking and possibly changing vote history for the location so a user might “re-vote.” As well, to avoid ***cartel*** or other ***malicious*** behaviors the number of votes and “revotes” per user should be limited.

### Location Notification

Content authors can include a location reference within the product details or content outline, this will enable a “Find Location” menu item on the phone. Choosing the option will launch the Location screens and will filter the location results view to only locations which are related to that product.

the application to the appropriate result list.

**DEV NOTE**: If locations in physical design are developed via tables and services the locationdetailsidref will need to be changed to type GUID or some backend ID scheme.

### Maintaining Location Detail Identifiers

Content authors can include a **locationdetails** and **locationdetailsidref** at the top of a text and picture topic page.

### Forms and uses of LocationDetails.xml documents

1. The first form is intended for use by system administrators, and would be used to create the location town view. It implements the TownRegion content model elements as well as provides references to a single Locationdetails subdocument *( see LocationDetailsSample1.xml).*
2. An optional form, the second form would be to provide a subset of Location model elements representing the locations within a Town/Region… (optimized in terms of contents as needed). *( see LocationDetailsSample2.xml).*
3. The third form would be used by location providers to submit location information and host full location detail information. This could be provided in segmented form for performance requirements. *( see LocationDetailsSample3.xml).*

The following is the model of **LocationDetails**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] version** | Optional simple float value representing a version of the same document |
| **TownRegions** | Optional 0-1 Group for TownRegions  |
| **Locations** | Optional 0-1 Group for Locations |

The following is the model of **TownRegions**:

|  |  |
| --- | --- |
| Name | Description |
| **TownRegion** | 1-99 TownRegion Elements |

The following is the model of **TownRegion**:

|  |  |
| --- | --- |
| Name | Description |
| **Name** | 1, 1-40 char Name of a Town or Region  |
| **LocationReferences** | 1 req. LocationReferences element |

The following is the model of **LocationReferences**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr]locationdetailssrc** | Optional URI as the source of the locations to be used as a navigation result set. |
| **LocationRef** | 1-99 LocationRef elements |

The following is the model of **LocationRef**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] inneridref** | Required string representation a Location id element within the same document |

The following is the model of **Locations**:

|  |  |
| --- | --- |
| Name | Description |
| **Location** | 1-99 Location Elements |

The following is the model of **Location**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] id** | Required. A unique ID for the Location |
| **Title** | Required 1, 1-60 char string represents the name of the location  |
| **Address** | Required 1, 1-60 char string for the address of the location |
| **Description** | Optional 0 to 1, 1-100 char string for the description of the location |
| **Telephone** | Optional 0 to 1, /d{3}-/d{3}-/d{4} pattern for phone number |
| **Website** | Optional 0 to 1, 60 character URI for website  |

# FlashcardGameContent

**FlashcardGameContent.xsd, FCG\_*XXXX*.xml**

This is a content document which authors a flash card user experience.

**User scenario**

When the user reaches a flashcard activity, the navigation behavior allows them to either use a the softbutton, or 1 key to flip and advance the cards:

Uses reaches a flashcard activity.

 ( first press ) flip the card flips to the backside
 ( next press ) the next card is shown
 [repeat until either they are out of cards ]
 Once out of cards if no link to another activity is defined ( previd )
 then the content outline is shown.
 [otherwise]
 present the activity defined at previd



The following is the model of **FlashcardGameContent**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] version** | Optional simple float value representing a version of the same document |
| **[attr]contentassetssrc** | Optional reference to a contentassets document ( defaults to contentassets.xml ) |
| **[attr]previdref** | Optional string reference to an content asset within a contentassets document |
| **[attr]nextidref** | Optional string reference to an content asset within a contentassets document |
| **TopicTitle** | Optional ( non displayed ) 1-60 character title string  |
| **Cards** | 1 group of Card elements |

 The following is the model of a **Card**:

|  |  |
| --- | --- |
| Name | Description |
| **(choice group)** | Required select 1 of Description or Image |
| **Feedback** | Req 1-60 character string for the text on the back of a card.  |

The following is the model of **Description**:

|  |  |
| --- | --- |
| Name | Description |
| **(self)** | Req 1-200 character string for the front of the card  |

The following is the model of **Image**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr]src** | Required URI for the local source of the image |
| **[attr]finalsrc** | (Form modified) relative path representing a folder/filename for the image |

# AdventureGameContent

**AdventureGameContent.xsd, AG\_*XXXX*.xml**: This is a content document which authors a scenario walkthrough experience.

**DEV NOTE**: an adventure should be all within the same content file – no more than 15 description and choice pairings will be in the same content file ).

Adventure begins displaying the first scenario found with the content document. The uses selects using up/down, soft selection ( aka a normal drill down list ). The scenario displayed then changes based on the idref used for the selected choice item. Adventures end, when no choices are provided ( a terminal scenario ).

They navigate to the next activity using menu or key selection.

The following is the model of **AdventureGameContent**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] version** | Optional simple float value representing a version of the same document |
| **[attr]contentassetssrc** | Optional reference to a contentassets document ( defaults to contentassets.xml ) |
| **[attr]previdref** | Optional string reference to an content asset within a contentassets document |
| **[attr]nextidref** | Optional string reference to an content asset within a contentassets document |
| **Title** | Req 1-22 character string.  |
| **Scenarios** | Required 1 group for 1-15 Scenario elements |

 The following is the model of **Scenario**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] id** | Required. ID for the scenario. |
| **Description** | Req 1-20 character string.  |
| **Choices** | Optional 0-1 group for Choice elements |

The following is the model of **Choice**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] idref** | Required. IDREF type leading to another scenario. |
| **(self)** | Req 1-40 character string.  |

# Sort Game Content

Provides a sorting experience providing a short list of terms/phrases and 2 categories to sort them by. It contains:

 Two category phrases (30c).

 1-10 terms (80c)

**User Scenario**

User starts the activity from a sort activity introduction page which lets the user know about the sorting activity ahead. Upon seeing the activity they are presented with a list, which animated down top to bottom and stop animating once the bottom item in the list reaches the blue highlighted agrea. They are presented with two areas below this which represent the categories the list needs to be sorted into.

They use a left or right button operation on their client to sort the list. A correct guess, highlights the word in green with a emphasized arrow on the correct category also appearing, the term also is removed from the list. Upon an incorrect guess, the term is highlighted in red and disappears – they will see it again so they can try again to resort them item.

Upon any guess, the user is present with the next item in the list - it drops down. And the scenario continues like this until all items are sorted correctly. Upon which they are presented with the answers screen which shows the categories and items under the appropriate category.



The following is the model of **SortGameContent**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] version** | Optional simple float value representing a version of the same document |
| **[attr]contentassetssrc** | Optional reference to a contentassets document ( defaults to contentassets.xml ) |
| **[attr]previdref** | Optional string reference to an content asset within a contentassets document |
| **[attr]nextidref** | Optional string reference to an content asset within a contentassets document |
| **TopicTitle** | Optional ( non displayed ) 1-60 character title string  |
| **Image** | 1 required Image element |

The following is the model of **Image**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr]src** | Required URI for the local source of the image |
| **[attr]finalsrc** | (Form modified) relative path representing a folder/filename for the image |

# Pan Image Content

Intended for larger or more detailed images such as line art or labeled screen shots, provides a 1:1 image presentation experience with panning functionality. This contains:

**User Scenario**

When a pan image activity is displayed to the user the entire content area is used to display portions of a lineart image at a 1:1 ratio. The user uses the softbutton or a combination of direction mapped buttons to pan the image in a cardinal direction, in increments ( such as a certain amount of move made upon button up ). A caption at the bottom informs them of the buttons to use for panning. These buttons are different than those used for menu and navigation actions.



The following is the model of **PanImageContent**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] version** | Optional simple float value representing a version of the same document |
| **[attr]contentassetssrc** | Optional reference to a contentassets document ( defaults to contentassets.xml ) |
| **[attr]previdref** | Optional string reference to an content asset within a contentassets document |
| **[attr]nextidref** | Optional string reference to an content asset within a contentassets document |
| **TopicTitle** | Optional ( non displayed ) 1-60 character title string  |
| **Image** | 1 required Image element |

The following is the model of **Image**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr]src** | Required URI for the local source of the image |
| **[attr]finalsrc** | (Form modified) relative path representing a folder/filename for the image |

# TipContent

**Tips.xsd, Tips.xml**:.

The tip content is used to present a tip or word of the day on the home screen.



Tips visibility happens under the following conditions.

If not courses are in the users My Courses list, then no tips appear.

(pseudo code)

haveCourseWithTips = false

tip = null

If courses are in the users My Courses list {

 for each course in the list {

 The course to pick the tip from is selected via
 randomization of a number from 1 to the total
 number of courses in the my courses list.

 haveCourseWithTips = determine if a tip content document exists

 if (haveCourseWithTips){

 tip = getTipContent(...)

 break;

 }

 }

The following is the model of **TipContent**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] version** | (default 1.0). version for document |
| **Tips** | Required 1-50 group of Tip elements |

The following is the model of **Tip**:

|  |  |
| --- | --- |
| Name | Description |
| **(self)** | Req 1-220 character string.  |

# OfferDetails

**OfferDetails.xsd, OfferDetails.xml**:.

Program Access offers are available from detail pages which have an associated program access component.



The following is the model of **OfferDetails**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] version** | (default 1.0). version for document |
| **ProgramAccessOffers** | Optional 0-1 list of ProgramAccess items |

The following is the model of **ProgramAccess**:

|  |  |
| --- | --- |
| Name | Description |
| **[attr] id** | Require unique id for the program access offering. |
| **ProgramName** | Required 40 characters for the program name |
| **AccessCode** | Require 60 characters for the program access code. |