Table of Contents

General Information

Platform Directory Structure and the "Common" Folder

The directory organization on the web server is shown right. Most content is unique to the lesson; however, some content and tools are shared among many — sometimes all — frames. Those files are located in the common folder.

Updates to Common Assets

The latest common folder is available on e2020 SharePoint here. You can also usually download a zip of the common folder and a zip of the templates, too. Although the SharePoint directory will update throughout the week, the zip file itself will be updated (at least) every Friday 5 PM EST combining all files in the common folder.

All users should update their local copies of this folder every week and make sure current work adheres to the latest build.

Linking to a Common Asset

Files in the common folder can be referenced using a relative path in this form: "`../../../../../common/folder-name/file-name.abc`" Note that there are five `../` before the word *common.*

Naming Conventions

All file names, CSS classes, and IDs should be named using lower-case letters and hyphens—no upper-case letters or underscores. (Some legacy elements may be exceptions.)

Protected Files

Please do not change the folders or files in **pink color**. They are part of the frame player infrastructure and changing them would cause major problem. If you think one of these files needs to be changed, email both Brian Duncan and Charlie Potter.

- frame-building-files *(May be named differently on the server)*
  - common
    - audio
    - calculators
    - closereader
    - flash
      - X *audio.swf*: Used to play frame audio
      - X *video.swf*: Used to play videos in older browsers.
    - images — *all common images, including icons*
    - lab-resources
    - math-interactives
    - passages
    - periodic-table
    - scripts — includes all widgets used globally
    - styles
    - templates
  - course 1 name * *[directory described in detail on next page]*
  - course 2 name
  - course 3 name

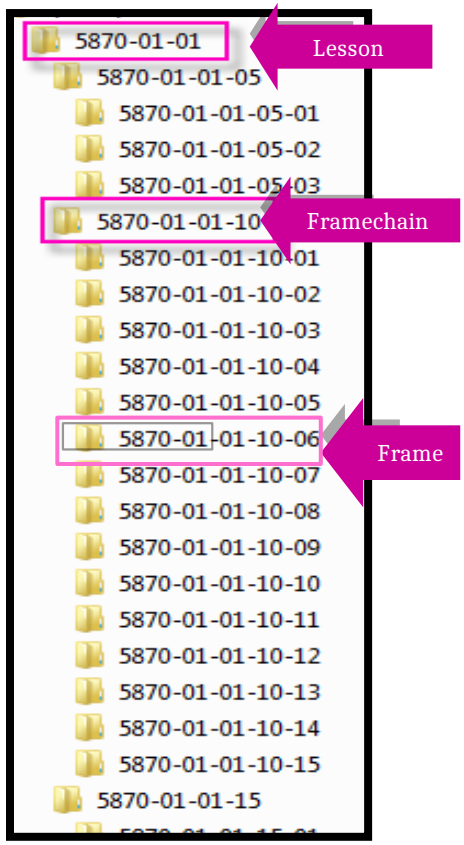
  [Additional Course Folders]

Never put files in the Course, Unit, Lesson, or Framechain level folders.

Be sure to read through this guide all the way.

Course Directory Structure

- course name
  - unit name *
    - lesson name *
      - framechain name*
        - frame name
          - [frame files]



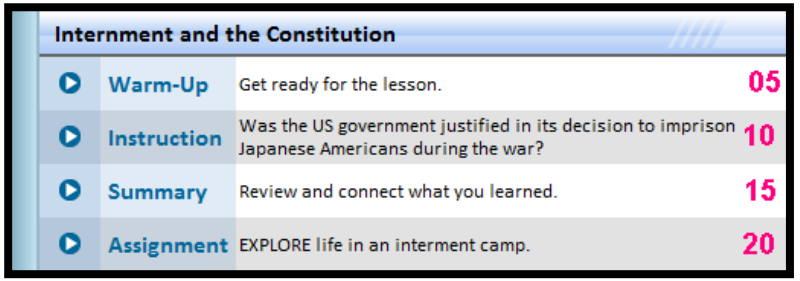
Naming Rules: Every folder and file is named according to the rules described below.

1) **Course MasterBuild ID (MBID):** For each course, the engineering team determines a 4-digit MBID, which never changes. Each course lives as a folder named according to its MBID. **Example: 5870**.

2) **Unit:** A standard course has 12 units, numbered from 01 to 12.

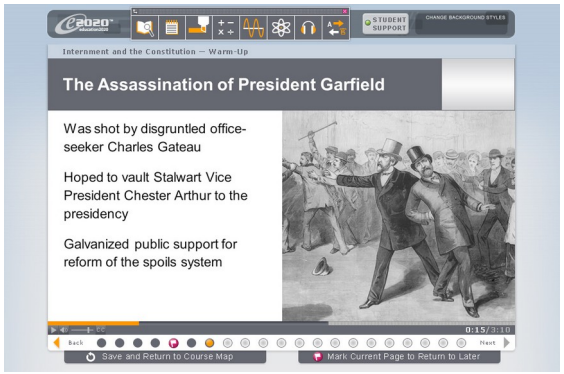
3) **Lesson:** A standard unit has 10 lessons. The lessons are numbered from 01 to 10. Each lesson lives simply as a folder inside the course folder. The units are numbered from 01 to 12, appended to the MBID.
Example: Unit 1, Lesson 1 of course 5870 lives in a folder titled **5870-01-01** shown below.

4) **Framechain:** A framechain is any activity in the course that is identified individually within the course structure. Framechains live simply as folders of frames. The framechain folders are numbered, starting with 05, and increase in increments of five. **Example:** the lesson shown below has four activities, each with its own framechain folder: Warm-Up (**5870-01-01-05**), Instruction (**5870-01-01-10**, highlighted above), Summary (**5870-01-01-15**), and Assignment (**5870-01-01-20**).



The four framechains above are the most common; however, other orders and additional assignments and instruction are common, too.

5) **Frame:** Each framechain is a sequence (“chain”) of individual frames (either videos or HTML pages) that students view in a frame player. (See snapshot shown below.)



Frames are numbered, starting with 01, and increase in increments of one. **Example:** The screenshot shown bottom right indicates 21 frames (via the small circles at the bottom of the screen). The pink flag marks the 6th frame, which is a video that lives in a folder titled **5870-01-01-10-06.**

The Frame Building Process

This guide describes how to build frames. Frames are created in HTML to allow flexibility and customized formatting. The process has 5 steps, each of which is detailed later in this guide:

Step 1: Enter Questions into Exam View

Step 2: Collect Images (including PowerPoint PNG Exports)

Step 3: Build the Frame's Folder

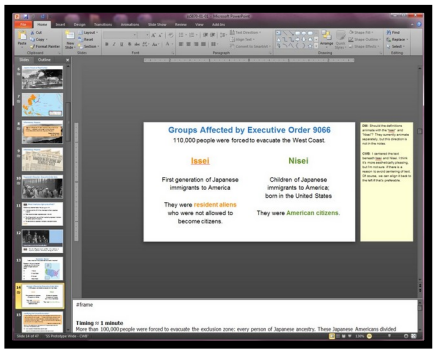
Step 4: Build the Frame's HTML Page with Templates

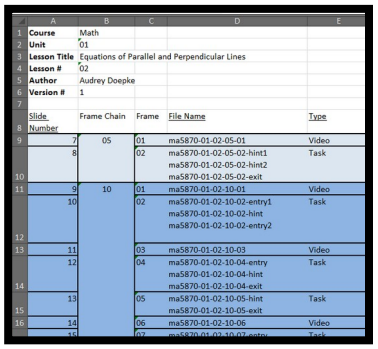
Step 5: Check your work.

Files Passed from e2020 Production to the Frame Building Process

The steps are all done after the production team finishes it work. For each lesson, the production team provides the following:

1) **A Lesson PowerPoint file (below left).** The lesson PowerPoint is the blueprint of the lesson. It guides the appearance of each frame and how the frame should function.





	A	B	C	D	E
1	Course	Math			
2	Unit	01			
3	Lesson Title	Equations of Parallel and Perpendicular Lines			
4	Lesson #	02			
5	Author	Audrey Doepke			
6	Version #	1			
7					
8	Slide Number	Frame Chain	Frame	File Name	Type
9	7	05	01	ma5870-01-02-05-01	Video
10	8		02	ma5870-01-02-05-02-hint1 ma5870-01-02-05-02-hint2 ma5870-01-02-05-02-exit	Task
11	9	10	01	ma5870-01-02-10-01	Video
12	10		02	ma5870-01-02-10-02-entry1 ma5870-01-02-10-02-hint ma5870-01-02-10-02-entry2	Task
13	11		03	ma5870-01-02-10-03	Video
14	12		04	ma5870-01-02-10-04-entry ma5870-01-02-10-04-hint ma5870-01-02-10-04-exit	Task
15	13		05	ma5870-01-02-10-05-hint ma5870-01-02-10-05-exit	Task
16	14		06	ma5870-01-02-10-06	Video

2) **A Lesson extraction sheet (above right).** This basic outline describes how the PowerPoint file breaks in into framechains and frames. It also includes any file names for video and audio files and identifies whether a slide is a video or a task (HTML page).

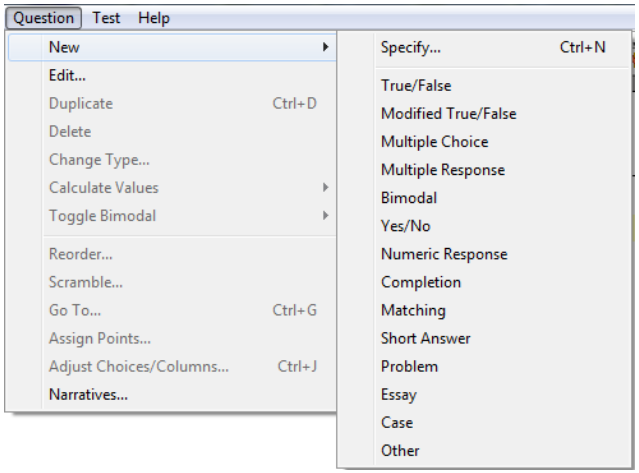
3) **Lesson Folder Structure:** The folders will generally be "pre-made" and follows the file structure described later in this guide.

4) **Other assets:** Teacher video and teacher audio is compressed and saved all together in a root directory and needs to be put in the appropriate frame folders. Other files, such as images, are generally either saved with the lessons OR must be exported directly from the PowerPoint file.

Step 1: Enter Questions into Exam View and Export

Most task slides have one or more questions for students to answer. To create a question, go to the **Question** menu, as shown below. Choose an option:

- **Multiple Choice** for Radio Button questions
- **Multiple Response** for Checkbox Questions
- **Essay** for Textfields, Textareas, NumberFields, and Drop-Down Menus. (This may be counter-intuitive because three of them are not actually essays, but this is how we do it.)



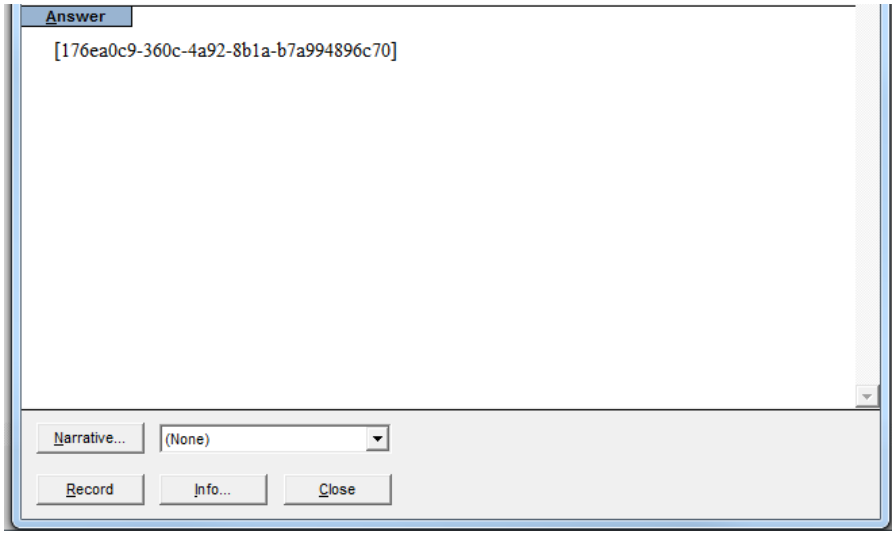
Other types of tasks are *not* done in Exam View, and instead are made up using special widgets. Examples include click-for-details, click-image, click-timeline, tile-sequence, tile, etc.

Information about QIDs

Every question needs a QID, or Question IDentifier. Our QIDs are in fact GUIDs ("Globally Unique IDentifier). To obtain a GUID, use any GUID generator, such as *this link*. The system uses the QID in several ways:

- It enables the question importer to save and track the questions. (The importer will not work without a GUID.)
- HTML pages display questions from the system by using a div that refers to the QID. (See later section of the guide.)
- The system tracks student performance by QID.
- You can edit a question by referencing the QID — either by remaking the question in ExamView or by editing the XML code directly. When you re-import the question, the system will overwrite the old question with the new one.

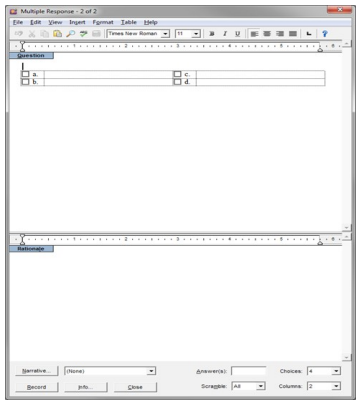
When making a question in ExamView, place it in the bottom box (labeled either Rationale or Answer, depending on the type of question), as shown below. Be sure to place the GUID in brackets, as shown.



How to Enter a Checkbox (Multiple Response) Question

Refer to the third question in the example test.

1) In Exam view, go to **Question > New** and choose **Multiple Response**. This will bring up the Exam View question editor for a Multiple Response question.

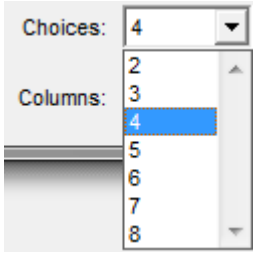


2) In the Question area at the top, put the question stem.

Question

Which of the following animals are mammals:

3) Change the number of answer choices using the "Choices" drop-down menu at the bottom.



4) Do NOT use the columns option in ExamView. Our system ignores it.

5) Type your answers in the area next to the corresponding checkboxes:

☐ a.	Dog
☐ b.	Platypus
☐ c.	Whale
☐ d.	Lizard

6) In PPT, correct answers are marked [X]. Select the correct answers in one of two ways.

a) Click on the checkboxes:

☑ a.	Dog
☑ b.	Platypus
☑ c.	Whale
☐ d.	Lizard

b) Type them in as a comma-separated list in the Answers section at the bottom:

Answer(s): A,B,C

7) Type the QID (any GUID) between brackets in the Rationale box. Copy the QID down somewhere for use while making HTML frames.

8) Type in the option tags you need after the GUID in the Rationale Box.

a) Type **NoABCD**. The NoABCD option makes a question display without the answer labels, such as "A", "B", etc. ExamView will show the labels, but our platform will hide them. ***Questions in frames nearly always use this option for multiple response and multiple choice questions, so be sure not to forget it.***

b) The PPT may indicate [-] for all answer choices. It may also say "survey" or "accept-any". In these cases, the editor wants any/all answers to be accepted. You should type **survey** next to the GUID. Unfortunately, ExamView won't save a question without an answer. Therefore, include is at least one (arbitrary) answer. When you upload the question, our platform will override the answer.

c) You can also type **randomize** in the rationale box after the GUID. The Randomize option specifies that the choices will be randomized when displayed to the student. Randomize should only be used in conjunction with the NoABCD option. (You still need choose a correct answer in Exam View.)

d) To place these options in a question, include them is the same area as the GUID. **Note that this method is only used for multiple choice and multiple response questions.**

Answer

[176ea0c9-360c-4a92-8b1a-b7a994896c70, survey, noabcd, randomize]

9) When you are done creating the question, click Record.

How to Enter a Radio Button (Multiple Choice) Question

For an example, see first question in the example test.

In Exam view, go to **Question > New** and choose **Multiple Choice.** This will bring up the Exam View question editor for Multiple Response questions. Follow the directions above for Checkbox questions. The only difference is that you must select just **one** correct answer. Again, always use NoABCD. If the answer

choices are all marked with an O, that means you should mark the question as a survey question. Otherwise, the correct answer will be marked !!!

How to Enter Drop-Down Menu, TextField, and NumberField questions

6) In Exam view, go to **Question > New** and choose **Essay**. Even though these three questions aren't really essays, this **is** the option you want!

7) In the question area, type the question text. To incidate a field or drop-down, type one of the codes below. (After import, the system will place the box or drop-down there.) Use a different number for each field or drop-down in the question. (Answers will be matched via the numbers.)

- Textfield (literal) questions use dollar signs $___1___$
- Drop-down questions use brackets [___2___]
- Number Field (numeric) questions use asterisks *__3__*

8) For textfield or numberfield, type additional underscores to control the size of the boxes. Using more underscores makes the boxes wider, which is better for students to type in. (This has no effect on dropdowns.)

9) Type the QID (any GUID) between brackets in the Answer box. Copy the QID down somewhere for use while making HTML frames.

10) Below the GUID, type the answer. First, type the question number in brackets. Below, type each individual answer on its own line below the brackets. (The PPT files might put answes for textfields in a list with commas, but you must separate the list into separate lines.) Textfield and numberfield questions can have more than one possible answer (any in the list will be accepted). Drop-down menu questions have just one correct answer with a * to the left. Example:

[1] – *indicates first question*
The answer to the first textfield or numberfield
Another possible answer to the first textfield or numberfield
Another possible answer to the first textfield or numberfield
[2] – *indicates second question*
Incorrect Answer
Incorrect Answer
*Correct Answer
Incorrect Answer

11) If the PPT file indicates that it is a survey question ("survey" or "accept-any"), place the option *survey* inside the bracket with the answer ID*:*

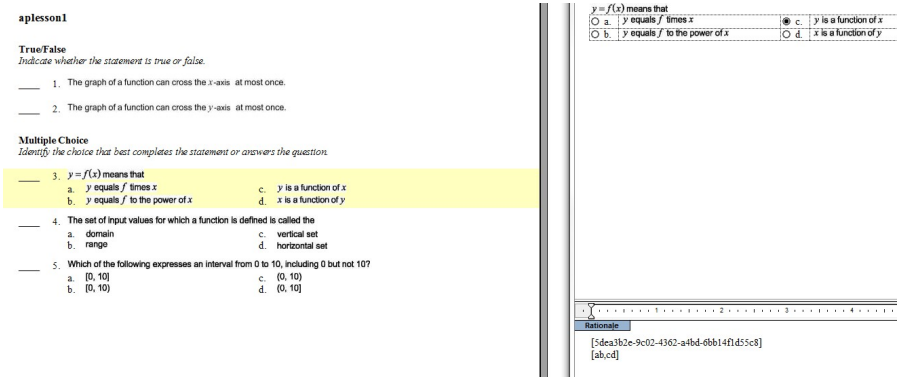
Example: [1, survey]

Unfortunately, ExamView won't save a question without an answer. Therefore, include is at least one (arbitrary) answer. When you upload the question, our platform will override the answer.

In the example test, #5 is a numberfield, #8 is a texfield, and #11 is a drop-down menu. The difference between textfields and numberfields is that numberfields only accept inputs that can evaluate to a number.

How to Enter Layout of Questions

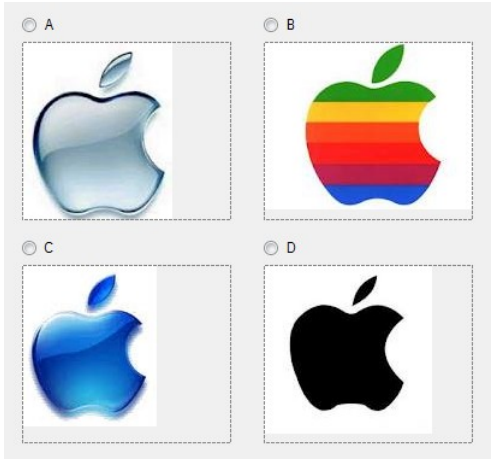
- Add a layout string right below the guid. Shown here as "[ab,cd]":



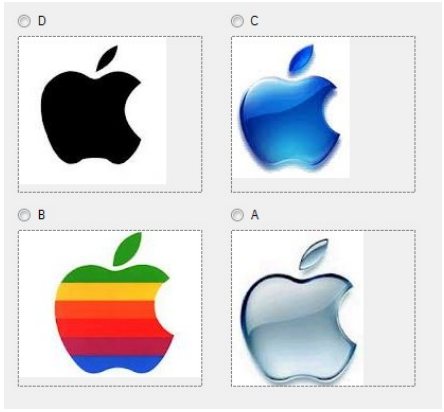
Format of the Layout string:

The comma represents a break between questions and order matters, therefore:

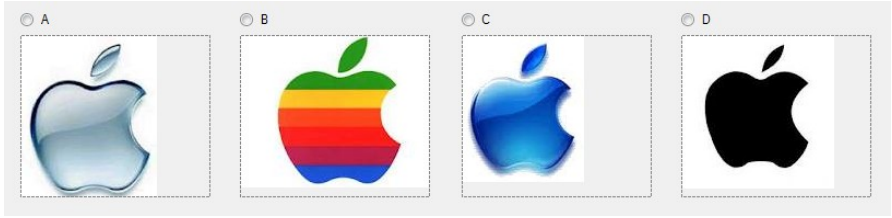
[ab,cd]



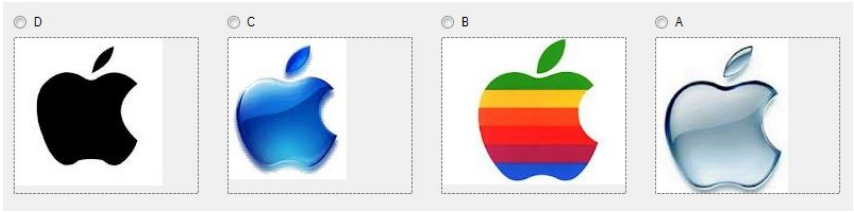
[dc,ba]



[abcd]



[dcba]



[abc,d]

Rules for the layout of the questions on frame chains:

If there are an unequal number of actual questions to layout characters in a layout string, then no layout will happen. It will default to a single column of multiple options.

If there is no layout string then no layout will happen. It will default to a single column of multiple options.

If the random question draw is enabled then the ordering of the questions specified in the layout string will not matter, but the actual layout structure will be retained.

How to Enter TextArea Questions

1) In Exam view, go to **Question > New** and choose **Essay**.

2) Simply put the text of the question stem in the Question area at the top. By default, the system will place the textarea box after the question stem.

3) Type the QID (any GUID) between brackets in the Answer box. Copy the QID down somewhere for use while making HTML frames.

4) Below the GUID, type [TA] using brackets. This code tells the system to put a textarea box for students to respond in.

5) Textarea questions can be graded on keywords. If the essay has keywords then the answers can be placed under it in the answer section just like a fill in the blank question. The list of key words should be newline-separated. Keywords can be more than one word. They are NOT case-sensitive. (The PPT files might put answes for textfields in a list with commas, but you must separate the list into separate lines.)

6) If the textarea is a survey question then the survey tag can be placed next to the GUID as with response questions.

7) The default height of a textarea in our platform is 100px. You can change the size by inserting the height you want into the [TA] tag. For example, if a text area needs to be 180px, the tag would look like this:

[TA height: 180px].

Use approximately 25px per line of text called for by the editors. (If this tag is malformed, the importer may apply the 100px default instead.)

Extremely Rare: If you want the textarea box to appear somewhere else, you can also type [TA] again in the Question area in whatever spot you want the text box to appear)

In the example test, #8 is a textarea question.

How to Export the file and Import the Question to our Platform

The process below generates two different files: an ExamView test file and a compressed folder that contains data used by our system (including an XML file with a .dat suffix). Each frame will have its own ExamView file. The number of questions in the ExamView file should match the number of questions on the frame.

1) Be sure you note the QIDs somewhere for later use.

2) Save the ExamView file using the frame's official name, as below:

*MBID-Unit-Lesson-Framechain-Frame**-task.tst***

3) Then, export the ExamView test into a form that can be read by the platform using the directions below.

a) In ExamView , go under File, Export and select Blackboard 6.0-7.0

b) Give a name to the exported file and click "save"

c) In the dialog box that pops up, you can type anything into the Name box, which isn't important. However, in the Directory Name box, type the name of the folder that any images will be stored in when the import is finished.

d) Once you're done with that you'll be given a .zip file named

*MBID-Unit-Lesson-Framechain-Frame**-task.zip***

4) Go to *http://qa.education2020.com/FileUpload* and upload your .zip file.

5) If the page responds **Files Successfully Upload,** the questions are saved on the platform and available to be placed into HTML.

6) To put the question into an HTML page, use the formula below, without any content inside the div. The API framework will automatically fill the div with the question when the HTML page loads in a browser.

```
<div qid="QUESTION ID GOES HERE"></div>
```

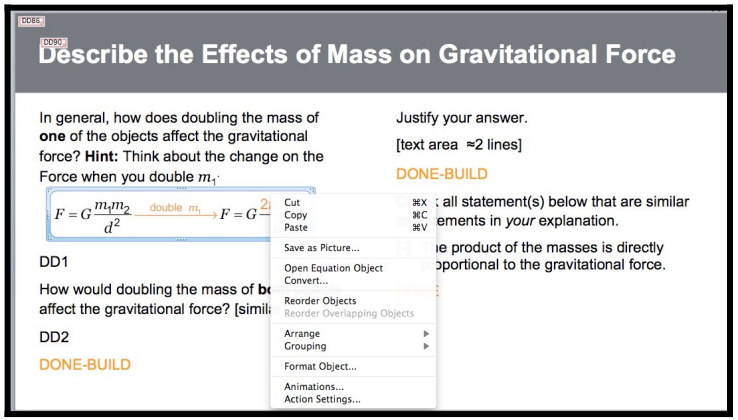
7) If changes need to be made, edit the .tst file and repeat the steps above. Alternatively, if you know how to edit the XML code, you can edit resource.dat (within the .zip) and re-upload just that file alone.

NOTE: If you are saving ExamView files that are used ONLY for assessments (not frames), the file name must follow the official convention for assessments (e.g. 6079-01-01-05). Files not named correctly will not appear in assessments!

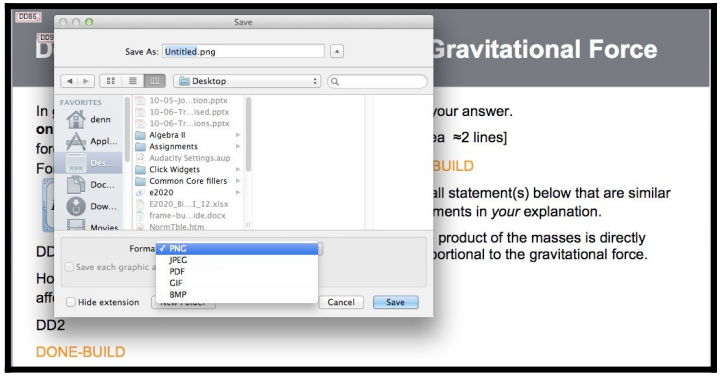
Step 2: Collect Assets

e2020 will provide all Adobe SWF, special interactives, and some images. Please embed them in the HTML as directed in the PPT files. However, other images, equations, and grouped artwork may be in PowerPoint only. You will need to save these as PNGs. Here's how to do that:

STEP 1: Right-click the image, equation, or grouped artwork. Choose "Save as Picture…"



Step 2: choose the PNG option and give it a good filename. **Never use PowerPoint's JPG option**; it looks terrible and uses the wrong resolution. PNGs export as 150px/inch. That is correct for our system.



Step 3: Finally, put that name in the HTML using a normal <img> tag. You may need to use size attributes and the style="float: left/right".

Step 3: Build the Frame's Folder

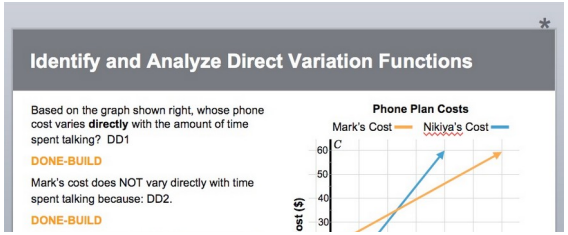
All elements of a frame are housed within the frame's corresponding folder. The lesson's folder structure will already have been built. The goal of this step is to move all files into the proper folders with the proper names.

Video Frames

For frames that are a teacher video, there will be no HTML page. Simply place the .mp4 (or .flv) file into the frame folder, and the software will know to play that video automatically. **Example:** the video file 5870-01-01-05-03.mp4 goes in in the folder 5870-01-01-05-03.

HTML Frames

(Also called *task frames*) When a frame is not just a single video, an HTML page must be created and stored in the frame's folder. The lesson's PowerPoint file guides how each HTML page should appear and function. HTML frames are indicated in PowerPoint by an asterisk at the top of the slide (see below) and are also identified on the lesson extraction sheet as a "task" or as "html". (We don't usually distinguish the two, even though some HTML pages will have no tasks.)



The files that are included in the folder will be determined by the PowerPoint slide. It is important to use the extraction sheet and PowerPoint when creating the task frames to ensure that all files are included. Below is a typical frame folder's contents. Note the naming convention:

*MBID-Unit-Lesson-Framechain-Frame)**-optional name.suffix***

Typical Files in a Frame

An HTML file	5870-01-01-05-03.html
Audio files	5870-01-01-05-03-entry.mp3 5870-01-01-05-03-hint.mp3 5870-01-01-05-03-hint.mp3 5870-01-01-05-03-click1.mp3 5870-01-01-05-03-exit.mp3
Images	See previous page about images. 5870-01-01-05-03-doggies.png random-name.png (not preferred, but it works fine, according to normal HTML conventions)
XML files for widgets	5870-01-01-05-03.xml [only one widget per frame]
Zipped Exam View File	5870-01-01-05-03-task.zip
[Regular] Exam View File	5870-01-01-05-03-task.tst
Folders	Generally, the frame folder does *not* contain other folders—and that includes images, which should be alongside the html. However, in special cases, it may be necessary.

Step 4: Build the Frame's HTML Page with Templates

If you wish, you can work directly with the HTML code and check your work by testing in a browser. However, you may also use Dreamweaver to build HTML pages. To Set Up a Site Using Dreamweaver Site:

- Unzip the latest common folder and the templates file to your desktop.
- Open Dreamweaver, click site > new site
- Name the site "*Frame_Content*" (or some other name) and point it to the unzipped folder
- Drag the pre-made course directory into the site.

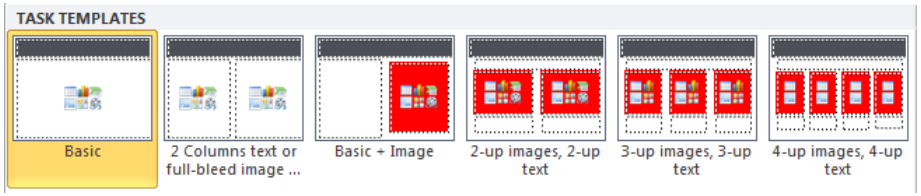
How to Add a Frame

- Click File > New
- Page from template > Frame_Content > Template
- File > Save As (This saves the new file so it will pick up paths to the styles and JavaScript framework)
- Create a new frame folder according to the directory structure described earlier. (Example: 1234-01-01-05-02).
- Save your file with the correct name (e.g., 1234-01-01-05-02.html)

The new file provides editable areas, one at the title at the top, and one or more for the body. Most frame content will go in the body section(s).

About the HTML Templates

There are HTML templates corresponding to each Task Layout that is in the Lesson Shell and Slide Templates PowerPoint presentation. Use these templates whenever possible.



There are also templates for CEE titles, Lesson Summary pages, and Science Lesson Summary pages for courses that contain Science Practice icons. Variations can be created using code described later in this guide.

Basic Structure of HTML page.

Here is the standard template for our HTML pages:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:mso="urn:schemas-microsoft-com:office:office"
xmlns:msdt="uuid:C2F41010-65B3-11d1-A29F-00AA00C14882">


<head>

<title>Isolated Frame</title>

<script type="text/javascript"
src="../../../../../common/scripts/jquery-latest.js"></script>

<script type="text/javascript"
src="../../../../../common/scripts/api_connector.js"></script>

<link href="../../../../../common/styles/Frame.css"
rel="stylesheet" type="text/css" />

</head>


<body>

<div class="title-bar">

      <h1 class="icon code">TITLE GOES HERE</h1>

</div>

<div class="content">

      content (including any columns)

</div>

</body>

</html>
```

Do not use a <header> tag. Instead, use with <h1> nested inside. We also no longer need a for "E2020 audio" or "FrameContent". Please do not use.

Title Bar (Frame Titles)

Without icon	With Icon
 <h1 class="no-icon">Your Title</h1> 	 <h1 class="icon-name">Your Title</h1>

Note: Spaces in icon names are separated by hyphens (lesson-question). Here is a list of icon names you can use.

no-icon	instruction	review
assessment	lesson-summary	think-about-it
assignment	lesson-question	try-it
assignment	objectives	warm-up
career-connection	overview	wrap-up
example	quick-check	
genre	real-world-connection	

Content Areas

There are templates for all of these. However, for reference purposes, the table below summarizes the different configurations of 1-4 columns.

Full-Width Content, no Columns Below

```
<div class="content">
Your content here
</div>
```

The below shows how to do two columns when there is NOT full-width text above the columns (i.e. between the title bar and the columns).

Columns

Two Column no Divider

```
<div class="content">
  <div class="left-column">
  Your content here
  </div>
  <div class="right-column">
  Your content here
  </div>
</div>
```

Use `<div class="left-column-separated">` instead if you want a divider line between the 2 columns.

Two Columns *AFTER* full-width text

```
<div class="two-up">
  <div class="left-column">
   Column 1 content here
  </div>
  <div class="right-column">
   Column 2 content here
  </div>
</div>
```

Three Columns

```
<div class="three-up">
  <div class="col-1-3">
   Column 1 content here
  </div>
  <div class="col-2-3">
   Column 2 content here
  </div>
  <div class="col-3-3">
   Column 3 content here
  </div>
</div>
```

Four Columns

```
<div class="four-up">
  <div class="col-1-4">
    Column 1 content here
  </div>
  <div class="col-2-4">
    Column 2 content here
  </div>
  <div class="col-3-4">
    Column 3 content here
  </div>
  <div class="col-4-4">
    Column 4 content here
  </div>
</div>
```

Plain Title slides (for Career Ed / Electives Courses)

Without Icon	With Icon
 Lesson Title	 **Lesson Title** **Lesson Question**

Text Classes and IDs

All text in the content areas should be within the paragraph tag.

Colored Text	All colored text (like vocabulary) should also be bold. Options for the *color* attribute include *orange, blue, green, gray, purple,* and *pink.* **Text**
Bullets	To obtain the standard, square, orange bullets with a hanging indent, do NOT use the tag. Instead, type: **Text**
Indented Bullets	**Text**
Summary Text	For the slide listing objectives at the end of the lesson summary, the content area will begin with <div class=”content summary”>
Science Practice Objective Icon	The easiest way is to use a template, but here is the way to do it without: Science Practice: **Text here**
Orange Sub Header	This is rarely used, but available. **Text**

Reading Panes

Editors will call for these in the Powerpoint by writing “Reading pane.” They may also indicate scrolling and color. The easiest way to obtain a reading pane is to use the template. If you need to fix code, the example below shows the pattern. You can replace the word **color** below with **orange, blue, green, gray, purple, pink, or white.**

```
<div class="reading pane-color">
<p class="hi">&quot;Paragraph with quotes&quot;</p>
<p>Text without quotes</p>
<p class="attribution"><em>-Title of text</em>, Author</p>
</div>
```

Scrolling Rules

Frames should NEVER scroll — WITHOUT EXCEPTION. However, reading panes, which are isolated boxes *within* the frame, can scroll. To obtain scrolling in a reading pane, simply set a height to the div.

```
<div class="reading pane-color"
style="height:##px">
<p>&quot;Text without quotes&quot;</p>
<p class="attribution"><em>-Title of text</em>, Name</p>
</div>
```

Note: If the reading pane is larger than the content area, a scroll bar will appear in the content area. That is undesirable, so please avoid.

How to Embed Images

Please put all lesson images in the same folder as HTML files, not in separate image folders. Except for specific instructional purposes, images should be on the right side of text, generally in the right-column div. Incidental images within text will be handled by the content divider classes and need no special coding.

For each image please include a simple description of the artwork or image that will sound logical to a hearing-impaired person.

Full-bleed image

There is a template for this. In case you need to fix code, the height *must be* 360px and the width is 372px or less. *Never* use attributions or captions.

```
<div class="right-column">
<div class="image-full-bleed">
      <img src="image.jpg" width="372" height="360"
      alt="image description for vision-impaired
      students" />
</div>
</div>
```

Image with margin, no attribution

There is a template for this. In case you need to fix code, the max width is 345 px and the max height is 318px. Smaller sizes are okay. Images may appear incidentally within the text of a slide and paragraphs can be before or after the image code.

```
<div class="right-column">
      <p>
      <img src="image.jpg" width="345" height="318"
      alt="image description" />
      </p>
</div>
```

Image with margin and attribution

There is a template for this. In case you need to fix code, the max width is 345px and max height is 300 px. Smaller sizes are okay. A lesser height is required for attributions longer than one line.

```
<div class="right-column">
    <p>
    <img src="image.jpg" width="345" height="318"
    alt="image description" />
    </p>
    <p class="image-attribtution-text">image
    attribution text</p>
</div>
```

Image with margin and caption

There is a template for this. In case you need to fix code, the max width is 345 px and max height is 300px. Smaller sizes are okay. A lesser height is required for attributions longer than one line.

```
<div class="right-column">
    <p>
    <img src="image.jpg" width="345" height="318"
    alt="image description" />
    </p>
    <p class="caption-text">caption text</p>
</div>
```

Adjusting Image Margin and Wrapping

If an image appears on its own or as a paragraph with no wrapping text, the paragraph tag ** ** should surround the image.

Exception: A full bleed image or an image that needs text to wrap around it should never be within a paragraph tag.

Entry and Exit Audio

To add entry or exit audio to a frame, simply include an mp3 file in the frame that uses the same name as the frame html file but adds "-entry" or "-exit" to the name. No editing of the html file is necessary; the API framework will pick up these files automatically. For example:

- 1234-01-01-05-01.html
- 1234-01-01-05-01-entry.mp3
- 1234-01-01-05-01-exit.mp3

fstacks

Frames use an object called a "frame stack" to organize, monitor, hide, and reveal content. The general "fstack" structure is shown below:

```
<div fstack>
    Question(s) with DONE button OR
    1 Custom Task (you cannot have both in the same
    fstack). See next page about custom tasks, which
    are rare.
    AND/OR other HTML
</div>
```

fstacks with Questions

Exam View Questions: To embed Exam View Questions (radio button, checkbox, drop-down menus, textareas, number fields, and textfields), follow the directions in Step 1 of this guide to enter the questions into ExamView and import them to our platform. Be sure to note the QID.

To put the question into an HTML page, use the formula below, without any content inside the div. The API framework will automatically fill the div with the question when the HTML page loads in a browser.

```
<div qid="QUESTION ID GOES HERE"></div>
```

DONE Buttons

Put a DONE button at the bottom of all stacks with questions—even if the editor forgot in the PowerPoint. DONE buttons may also appear in "Empty Tasks" (see below). Copy and paste the text below to get a DONE button. ***NOTE:*** *the div could be a span instead if you want the DONE inline.*

```
<div fdone class="done" title="done"
onclick="API.Question.attempt(this);"
onhint="API.Audio.playAudio('hintfilename.mp3');"></div>
```

HINTS: The code above includes a call to play a hint audio. Each hint audio is tied to a DONE button, and the hint audio plays if students fail *any* question in the stack. **Please provide the complete file name.** (The file generally lives alongside the HTML file.) If a stack does NOT have audio, leave out the "onhint" attribute in that case.

"Empty" Tasks

Sometimes, a stack will contain just HTML—no question or custom task. That's fine. Code it as shown below.

```
<div fstack>Any HTML + optional done button.</div>
```

Task Stack Building

PowerPoint files often call for **DONE-BUILD**. The purpose of this is to hide later fstacks, revealing them after students complete earlier fstacks. To specify that a stack will start as hidden and then appear ("**BUILD**") after a student completes the previous stack, use the ***display:none*** style, as below:

```
<div fstack style="display:none;">Questions & a
button go here</div>
```

If there is Transition Audio, put its file name in the DONE button like this:

```
<span fdone class="done" title="done"
onclick="API.Question.attempt(this);"
onhint="API.Audio.playAudio('hintfilename.mp3');"
oncomplete="API.Audio.playAudio('transitionaudio.mp3');">
</span>
```

The system will automatically reveal the hidden fstacks in the order of appearance within the HTML. Therefore, never put tasks in a table and expect them to build down the column.

Other Content within Frames

fstacks with Custom Tasks (Non-Question Tasks)

An fstack can also contain a Custom Task *instead* of questions. (Never both!) Custom Tasks are generally either SWFs and some DHTML widgets that ***must*** be completed by a student before moving on. (Use an empty task for simple animations and most widgets.) The PPTX must provide the taskID (a GUID) for the Custom Task. Code the HTML as shown below:

```
<div fstack>
   <div taskid="GUID GOES HERE">
         code for the task (Can NOT be an iframe)
   </div>
</div>
```

Notes: The programmers who make the Custom Task will visit *this link* and obtain a GUID to use as the taskID—a unique ID for that task so that student's completion can be saved by the system. That link MUST be put into the PPTX file for the person making HTML page. That code must also be called in the object's code using API.Frame.completeTask(guid)

Incidental Audio

To embed a link to play audio, please use the formula below, using a tag instead of an <a> tag:

```
<span class="span-link"
onclick="API.Audio.playAudioEx('audio1.mp3')">link
name</span>
```

Links to the Close Reader (Long texts)

To provide a link to the close reader, type the following, replacing the ### signs with the correct height and width needed.

```
<a href="link.html" onclick="window.open(this.href,
'width=800,height=#'); return false;">link text</a>
```

The HTML file for the text will automatically look in the common folder to find the javascript needed to run the close reader tools.

Calculators Embedded Right on Screen

The matrix and regression calculators are both embedded as iframes. There is a template for these. Please use the template! Its

location is common/calculators/html-templates (for calculators)/frame/template-with-regression-calc-embedded.html

Link to a Calculator in a Pop-Up Window (i.e., not embedded)

To provide a link to the graphing calculator, type the following, replacing the ### signs with the correct height and width needed.

```
<span class="span-link" onclick =
"window.open(
'../../../../../common/calculators/graphing-
calculator.html','','height=###, width=###')" >link
text</span>
```

You can also type *matrix, statistics,* or *regression* instead of *graphing* to access the other calculators.

Links to the Periodic Table

periodic-table.html: To provide a link to the periodic table, type the below. Editors *may* ask for a specific element (replace ## with the element's atomic number) and/or tab number (replace the # with 1, 2, or 3 depending on whether they want the Element Class, State of Matter, or Orbital Properties tabs). Both may be omitted.

```
<span class="span-link" onclick="API.pop-up
('../../../../../common/periodictable/periodictable
.html?id=##&tab=##')">link name</span>
```

Element-info.html: To provide a link to a description of a specific element from the periodic table, type the below. The editors must ask for the specific element (replace ## with the number), which is the element's atomic number:

```
<span class="span-link" onclick="API.pop-up
('../../../../../common/periodictable/
elementinfo.html?id=##')">link name</span>
```

JavaSketchpad Applets

Simply copy and paste the code provided in the PowerPoint by the math writers into column 2. Change the .jar file to have the path below:

```
../../../../../common/scripts/jsp5.jar
```

HTML iFrames

There are occasionally other things that can be embedded into the frame, such as special videos or special applets. You can use an iframe for this. However, keep in mind that nothing in the iframe can "communicate" with the frame — so questions or tasks can't go there—only "passive" content.

Links, General information

Specific examples for how links should be written in html are outlined in the previous sections. If they are not described above, the following are the general guidelines for how links should be coded. Consult with the engineering team to determine if the link falls in either of these two categories:

Links to external content (ie: not hosted or created by education 2020), or internal content that requires the tool bar (like the CloseReader), should use an **a href** tag. See *Links to the CloseReader* for an example.

For links to internal content (calculators, tools) that do not need the tool bar, use a span tag with an onclick attribute. See *Links to a Calculator in a Pop-Up* window for an example. Be sure to use the "span-link" class so that the span-based link is formatted correctly.

When using the onclick approach with an image, such as for audio and DONE buttons, do **NOT** include the class "span-link". Likewise any code snippet that already includes a class should **NOT** use the "span-link" class.

Click Widgets — Documentation

Introduction

This document is a resource for anyone producing a slide designated for one of the widgets listed below:

- Click-Links
- Click-Images
- Click-Timeline

Adding a Click Widget to an HTML-based Frame

The widget can be added to an HTML page by placing one of the following codes into the body of the page beneath the frame's title:

```
<script language = "JavaScript"
src="../../../../../common/scripts/links-and-
content.js"></script>
```

```
<script language = "JavaScript"
src="../../../../../common/scripts/click-image.js"></scri
pt>
```

```
<script language = "JavaScript"
src="../../../../../common/scripts/timeline.js"></script>
```

File Types and Folder Structure

- All file names and suffixes need to be in lower case.
- All images must be JPG format unless where otherwise specified.
- All audio must be in MP3 format.
- **File Naming:** The paths to all associated files are generated based on the name of the HTML page the widget is inserted into, and files should be named accordingly. For example, for an HTML page named 0001a.html:

  The XML file must be named 0001a.xml

  - JPG files for the Image-Links or Timeline must be named 0001a-image1.jpg, 0001a-image2.jpg, 0001a-image3.jpg, etc. Links-and-Content images should be named the same way, for convenience.
  - MP3 files must be named 0001a-audio1.mp3, 0001a-audio2.mp3, 0001a-audio3.mp3, etc. (Entry and Exit audio are named as usual.)
  - Other files not automatically linked to by the widget (GIFs and other images hand-coded into the XML, etc.) should be named according to usual file-naming convention.
- Widget files should be placed in the course directory as described below:
  - The widget JS file is kept in the ***common*** "scripts" folder.
  - XML file is kept in the lesson's directory adjacent to the HTML file.
  - JPG and other images should be kept adjacent to the HTML file.
  - MP3 files should be kept in the audio folder

Images and Image Sizes

- In the image-links and timeline widgets, link images are automatically accessed and inserted by the widget's javascript based on the number of `<alink>` tags in the xml file. The <alink> tag only includes caption text, not a link to the image itself, which is generated automatically.
- **Size of Images**

  Image-Links: Though an indefinite number of images can be added to the links area, in order for them to fit the frame area, image widths must be (744/*x*) – 10 pixels (where *x* = the total number of images). Image height can be anything that still accommodates the display area, as defined by the AP in the linksAreaSize variable of the xml document.

  Timeline: Images on the timeline itself are resized versions of the display area images. In order to appear on the timeline, an appropriately-named full-sized image must exist in the images folder. The full-sized image may be of any dimension, provided they fit in the display area and with consideration that they will be resized to constrained proportions based on their height to 70px tall in "top" mode and 50px tall in "left" mode. Full sized images are automatically added to the display area and are aligned either to the left or right, based on the `align` attribute of the `<alink>` tag in the xml. Though display area images can be of any dimensions, the recommended size is 210 x 210 pixels.
- **Background Image for the entire frame:** Accomplish as you would any background

- **Background Image for the timeline line:** You can add a custom background image to your timeline by adding the name of your custom image in the images folder to the `timelineImage` varable in the xml document, i.e. `timelineImage="timeline.jpg"`. Leaving the variable blank will default to the standard blue line. The timeline image can be any format, including jpg, gif or png and may include transparencies or animation. The image must be 744 x 30 pixels in "top" mode and 30 x 360 pixels in "left" mode.
- Images used as links default to blue link border with orange rollover states. Should alternative link colors be necessary, paste the following styles into the head of the html document (below the css link) and customize the color codes as needed:

```
<style>
a:link img {
	border: 2px solid #CC0000;
}
a:hover img {
	border: 2px solid #FF0000;
}
a:visited img {
	border: 2px solid #FF3366;
}
a:active img {
	border: 2px solid #FF0033;
}
</style>
```

- **Design Rule:** For Image-Links and Timeline widgets, avoid incidental images in the content area, lest the layout appear cluttered or awkward.

MP3s

- MP3 files are automatically called by clicking a link, based on the order of the link clicked (first link calls "audio1.mp3", second link calls "audio2.mp3", etc). A missing MP3 will result in no audio being called.
  - **Design Rule:** If there is an audio file assigned to one link, there must be audio files assigned to all links. Otherwise, all links must have no audio associated with them.
  - Presently, clicking a link with a missing MP3 will result in the frame's audio player breaking. This has been logged as a bug by the developers.

XML structure and elements

The page XML will have the following format:

```
<xml>

<defaults />

<intro>Introduction text here.</intro>

<alink>Link text here</alink>
<content>Display area text here.</content>

<alink>Link text here.</alink>
<content> First link content goes here.</content>

<alink>Link text here.</alink>
<content> Second link content goes here</content>

</xml>
```

- All xml must be within the `<xml></xml>` tags and any HTML must follow strict XML tagging, i.e. opening and closing tags or self-ending tags (e.g., `<br />`)
- `<defaults />` must be terminated with a forward slash before the last bracket, and contain the variables that will be passed to the widget (not all of the variables below are used in all widgets):

- `linksAreaSize` (**links-and-content** and **image-links**) is the height of the links area in pixels. The frame player's total available space for the widget is 360px tall. A linksAreaSize of `"0"` means the links area will not appear, and a linksAreaSize of `"360"` would mean no display area would appear. The default size is 150 pixels. There is no need to add "px" to the variable.
- `linksPosition` (links-and-content and timeline) sets the format of the widget based on the position of the links area. It can be set to `left`, `right`, top or bottom and defaults to left. (Timeline uses only top or left and defaults to top).
- `timelineImage` (**timeline** only) identifies the image file for use as a custom timeline background in the images folder, for example:
  <defaults timelineImage="timeline.jpg" />
  If the variable is left blank, the timeline defaults to a four pixel blue line.

- `<intro></intro>` is an optional tag that contains the text to an introduction that fills the display area when the widget first loads. If no introduction is necessary, remove the tags entirely and the display area text will default to the `<content>` of the first link.
- Entry audio will play as usual when the `<intro>` tag exists in the xml, but when an intro does not exist, audio should default to the page's –audio1.mp3 file. This functionality should be incorporated by the developers.
- `<alink></alink>` tags defines the position of the link images and contains the caption text displayed over them. If an image link does not have caption text, the `<alink></alink>` tags still need to be in place, but without text between them.
  - The align attribute (**timeline** only) can be set to "right" or "left" and sets the alignment of the full sized image in the display area.
  - The pos attribute (**timeline** only) positions the images on the timeline. In "top" mode, the pos attribute is a percentage of the distance between the left and right ends of the timeline as expressed as a fraction of 1. A pos value of 0 places the image at the left end of the timeline. A value of 1 places it at the right end of the timeline. Values of .25, .3, .5, and .75 place images at 25%, 30%, 50%, and 75% from the left, respectively. The same goes for the timeline in "left" mode from top to bottom.
- The order in which `<alink>`'s are listed defines how the link's assets are named. For example, when the first instance of `<alink></alink>` is clicked, it calls the text of the first `<content></content>` tag, and calls the `page_name-audio1.mp3` audio, if those assets exist.
- `<content></content>` tags contain the text to be displayed in the display area upon clicking the corresponding link (as mentioned above). It can contain tags for breaks and paragraphs as well as styles inside and elements .
  - <displayImage /> (**links-and-content** only) can be placed anywhere within the text of the main `<caption>` contents to define where the image will be inserted.
    - align is an attribute that defines the position of the image inserted with the <displayImage /> tag. It can be set to right, left or center as shown below:
      <displayImage align="right" />

      Setting align to "left" or "right" wraps the content text around the image, while setting align "center" will center the image without wrapping the text.
  - `<subcap></subcap>` and `<supercap></supercap>` (**links-and-content** only) are tags contained in the `<content>` text to define captions above (supercap) or below (subcap) the image when the displayImage is also being used in the `<content>` area. An image may have either a `<supercap>` or a `<subcap>`, but never both. To display, the tags must be the first text appearing after the opening `<caption>` tag, like below:
    <caption><subcap>Caption text here</subcap> Main display area text goes here. <displayImage align="right" /> Additional text after the image goes here.</content>

    Design rule: while caption text should not exceed the width of the image, in cases where longer captions are necessary, the captions will not automatically wrap, and lines must be manually broken with a `<br/>` tag.

General rules and notes

- When all links are clicked, the widget sends an end point signal to the frame player telling it to allow the student to proceed to the next page.
- The click-image widget is compatible with IE, Firefox, Chrome and Safari.
- Due to the blue link border around any link images, images should avoid being designed to fit the rectangular area and avoid excess white space or unconventional image edges.
- **Design rule:** Title text inside the display area should be styled inline with `<b class="orange"></b>`
- In image-links and timeline widgets, `<alink></alink>` tags are necessary for the proper placement of images and must be included even if no text is placed between the tags.

Appendix A: API Framework Functions

API.Frame.complete()

Call this function when whenever the user has completed all tasks in a frame to mark the frame complete. If the frame contains questions, the database will ensure that the user has completed all of their work before allowing the frame to be marked complete.

Important: The framework already handles task flow, so this should be used only for custom pieces.

API.Frame.completeTask(guid)

This function should be called by a non-question task (i.e. widget / interactive) when completed by the user. The guid passed must be the same one described in the taskid attribute of any frame using the task.

If the custom task is loaded in an iframe, then a reference to the API must be obtained before this call is available. A reference to the API can be obtained through use of the scorm 1.2 standard API discovery algorithm (see for more info: *http://scorm.com/scorm-explained/technical-scorm/run-time/api-discovery-algorithms/*).

API.Question.attempt(this)

Place this call in the onclick event of all stack done buttons. This will allow the framework to find the button clicked and change its display to properly show the user feedback on the status of the stack. Be sure to provide the parameter ‘this’.

API.Audio.playAudio(filename, callback)

This function can be used to play mp3 audio files. If given a filename parameter that is not an absolute path, the filename will be interpreted as an extension of the name of the frame (e.g. if the frame’s name is ‘5870-01-01-01.html’ and the filename parameter provided is ‘hint’, then the function will attempt to load the file ‘5870-01-01-01-hint.mp3’ from the local folder). The callback parameter is called on completion of audio playing or upon failure of the audio file to load. While audio is playing, the user will be unable to interact with the window.

API.Audio.playAudioEx(filename, beginCallback, endCallback)

This function can be used to play an mp3 audio file without preventing the user from interacting with elements on the screen. Avoid using playAudioEx unless you're attempting to override the screen locking of API.Audio.playAudio. You also can add two parameters to the above — beginCallback and endCallback, which are optional, and if passed, will be called when the audio begins playing and ends respectively.

The callbacks are function references (and are optional). The function references here are the results of a lambda function.

```
API.Audio.playAudioEx("entry", function(){console.log("The audio has begun playing"), function(){console.log("The audio has completed")}});
```

This example shows how this would be done using named functions:

```
function aBeginCallback() { console.log("The audio has begun playing"); }

function anEndCallback(){ console.log("The audio has completed"); }

API.Audio.playAudioEx("entry", aBeginCallback, anEndCallback);
```

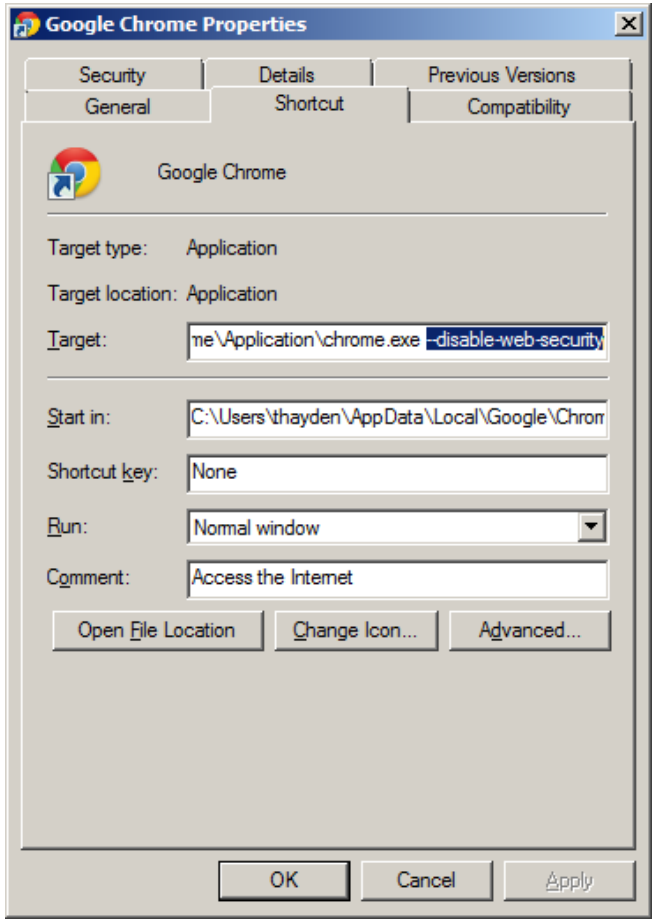
These callbacks can be used to do any number of things such as displaying some kind of information to the user while the audio is playing, or marking the frame complete once the audio has finished.

Appendix B: How to View on a Local Machine

How to Load Frames on a Local Machine

Chrome will not load questions when testing locally. This is due to a default security setting in Chrome that we are not able to get around at this time. Because of this, testing the frame in its entirety is only possible once Chrome has been started with a special flag.

To preview frames when testing locally, you will have to set the startup flag: --disable-web-security

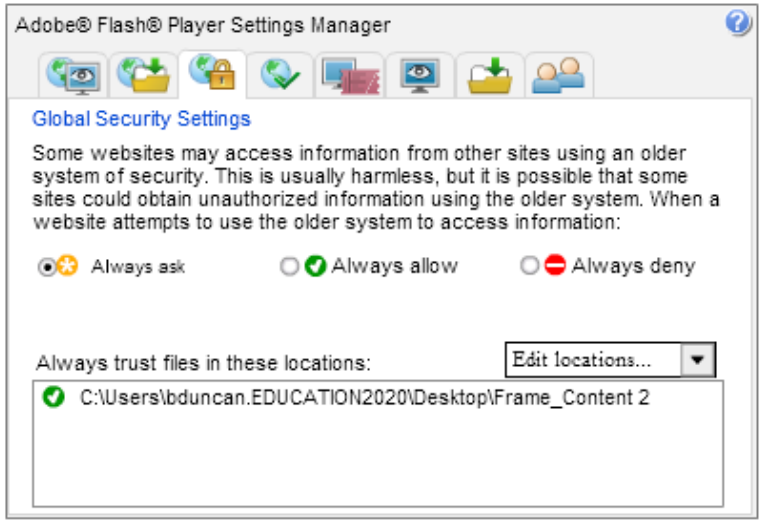


How to Play Back Audio Files on a Local Machine

Internet Explorer 7 will not load questions when testing locally. This is due to security settings in IE7 that we are not able to get around at this time. Because of this, testing the frame in its entirety is only possible once it has been published to the e2020 servers.

To preview audio when testing locally, you will have to set Flash security settings. Go to the site: *http://www.macromedia.com/support/documentation/en/flashplayer/help/settings_manager04.html*

Add your working folder to the “Global Security Settings” section under “Always trust files in these locations”



Appendix C: Scraps

The most common configurations are 1:

☐ a.	
☐ b.	
☐ c.	
☐ d.	

And 2:

☐ a.		☐ c.	
☐ b.		☐ d.	