Summary

The document has 4 sections, summarizing the requirements for MLN web-based training (WBT) courses to operate in the CMS.gov environment and standardized content to support learners interacting with the WBT in a stand-alone environment.

Section 1: Functional Changes

- Removal or deactivation of SCORM calls
- Course menu functions as dashboard, has record of progress, and includes certificate button
- Generation of certificate

Section 2: Technical Specifications

- Technical specifications and sample implementations for Section 1 items

Section 3: Content Changes/Specification

- Changes to wording in the introduction page, help page, and pass/fail pages in the assessment
- Generating the certificate of completion within the courseware
- Post-assessment is now referred to simply as Assessment

Section 4: Packaging Requirements- Delivery of Courses

- File structure for deliverables
- Instructions for file load to CMS
- Instructions for URL

Notes:

Technical specifications in Section 2 are best practices that have been tested in the CMS.gov environment and provide guidance.

Section 1: Functional Changes

Removal of SCORM

The WBTs must run in a stand-alone environment that doesn't make calls to the SCORM API or rely on the API as a persistence layer for learner data. The courseware must implement services given by the LMS (but not SCORM).

Examples of these services include (but aren't limited to):

- Bookmarking
- Lesson status
- Assessment results
- Lesson window management (launching and closing lesson windows, usually handled by LMS [but not SCORM])
- Certificate generation (covered later in this document)

Course Menu & Navigation Changes:

, The courseware must deliver the following functionality around launching a course lesson:

- Present lesson menu
- Enforce lesson prerequisites
- Display and update status of individual lessons
- Manage and persist assessment scores
- Manage and present certificate
- Open and manage lesson browser window

The course must now take on these other responsibilities. By sharing a persistence layer between the course and the menu, all the other functionality is within reach.

Menus and Prerequisites
While the LMS created a dynamic menu based on the contents of imsmanifest, it's up to the course author to build a menu interface. This can either be data-driven (assuming there exists a data structure that imitates the contents of imsmanifest) or static.

The prerequisites function should be hard-coded as it's always the same:

- If lesson B has no predecessor, it may be attempted (launched).
- If lesson B has a predecessor A, then predecessor A must be completed before lesson B can beattempted (launched).

Certificate Generation
The Certificate is a PDF that needs to be generated in the browser, so we require a reliable suite of tools to generate and preview PDFs. IE11 and Legacy Edge don't need to be supported. A successful combination is:

- jsPDF v2.3.1 (PDF generation)
- pdfobject v2.x (PDF preview in browser; relies on browser for PDF rendering)

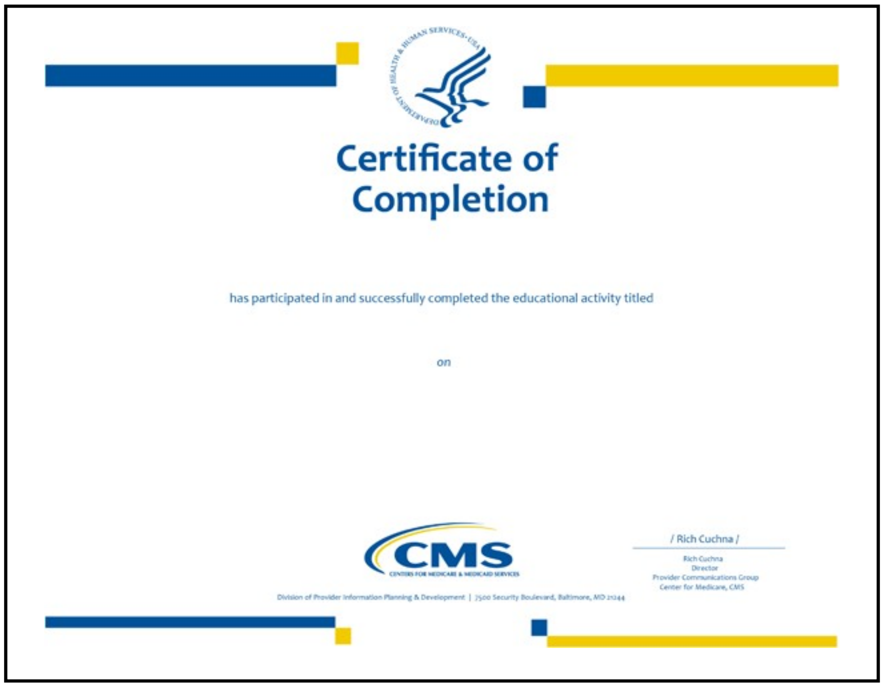
CMS mandates this library combination; it’s tested and confirmed as working in all supported browsers.The general learner flow is:

1. Collect learner name
2. Collect certificate date (LocalStorage) and course name
3. Optional: The collected data is trivially obfuscated with ROT13 to hinder certificate fraud
4. Pass collected data to PDF generator (for example, passed on the query string into an iframe housing PDF generator)
5. Generates a PDF using the jspdf library
6. Learner previews certificate in modal window
7. Learner has option to download PDF

Certificate Layout
All course completion certificates must display the following elements:

- Learner Name
- Course Name
- Date assessment was successfully completed with a passing score

All certificates must use the same layout for consistency across courses. See sample below. CMS has an image that can be used as a background image upon request. Contractors should contact their COR for certificate templates.



Certificate of Completion

has participated in and successfully completed the educational activity titled

on

/ Rich Cuchna /

Rich Cuchna
Director
Provider Communications Group
Center for Medicare, CMS

CMS
CENTERS FOR MEDICARE & MEDICAID SERVICES

Division of Provider Information Planning & Development | 7500 Security Boulevard, Baltimore, MD 21244

PDF Generation

Assuming the above combination suggested of PDF libraries, a PDF can be generated in a dedicated browser window (that's iFrame) that easily isolates it from the course.

The text strings written into the PDF can be passed either via LocalStorage or simply over the URL query string. These values include:

1. Course name
2. Learner name
3. Certificate Date (in format <full month name> <date>, <year>; no abbreviations)

There's a static background image upon which the all the text strings can be placed. The location is always the same, unless the background changes.

The developer must manage long learner and course names (line wraps).

Section 2: Technical Specifications

Removal of SCORM

It’s required for a client-side persistence layer to make learner data available across sessions. (Note: The courseware deployment has no authentication layer or server-side storage, so we won’t support synchronization of persisted data across devices.)

There are two capable candidate technologies already implemented in the browser:

1. **Cookies**
2. **LocalStorage API**

Solution 1: Cookies—It would be possible to serialize the entire learner state to JSON and save it into a cookie, but we would unnecessarily be sending all course-related cookies with each request to all page requests within cms.gov. Also, browsers limit the size of all cookies on a per-domain basis.

Solution 2- LocalStorage API - If a LocalStorage key is global to a course (instead of specific to a lesson), simply omit the lesson segment of that key. For example, assessment status (persisted across sessions) or learner-preferences (shared across lessons).

LocalStorage API—LocalStorage is a better fit given the limitations of using Cookies. For consistency, (and to avoid collisions with other courses), all keys must use the following naming scheme:

```
wbt.<course abbrev>.<lesson nr>.<key>
```

Example:

The World of Medicare course (WOM) has the following keys for Lesson 3:

```
wbt.wom.lesson03.lesson_status
wbt.wom.lesson03.lesson_location
```

You **must** use the following keys names for consistency across courses.

Lesson bookmark	wbt.<course abbrev>.<lesson nr>. lesson_location
Lesson completion status	wbt.<course abbrev>.<lesson nr>. lesson_status

It’s up to the course author to define a proper course abbreviation and lesson number for each course.

Route SCORM API Interactions by Façade

It’s recommended to route all SCORM API interactions through a façade (see Façade Pattern: https://en.wikipedia.org/wiki/Facade_pattern). This enables introspection of SCORM calls for debugging purposes, as well as pre-plumbing the course for interfacing with alternative persistence layers.

Use of the façade pattern trivializes swapping out the SCORM 1.2 implementation with an alternative that implements the SCORM API, but uses Local Storage for persistence instead of the LMS.

Sample Implementation of Façade
An example call to a façade implementation of LMSGetValue:

```
doLMSGetValue = function (name) {
  const lessonNr = getLessonNumber();
  const lessonName = getLessonName(lessonNr);
  let ret;

  switch (name) {
    case 'exit': {
      break;
    }
    case 'lesson_location': {
      ret =
localStorage[`wbt.${courseAbbrev}.${lessonName}.lesson_location`];
      break;
    }
    case 'lesson_status': {
      ret = localStorage[`wbt.${courseAbbrev}.${lessonName}.lesson_
status'`];
      break;
    }
    case 'lesson_mode': {
      break;
    }
    case 'mastery_score': {
      if (lesson.match(/([Aa]ssessment)/)) {
        ret = localStorage[`wbt.${courseAbbrev}.mastery_score`];
      } else {
        ret = '';
      }
      break;
    }
    case 'session_time': {
      break;
    }
    case 'suspend_data': {
      break;
    }
  }

  return ret;

}
```

Naturally, if the course isn't pre-plumbed with a façade in front of the actual SCORM API, the effort is larger. It may still be beneficial to add such a façade to ease any future migrations (for example, MLN decides to deploy to a SCORM or TinCan LMS in the future).

Course Menu & Navigation Changes:

Make a choice early on of how to open the lesson window. Below are 3 approaches, each with pros and cons:

FOR MIGRATING EXISTING COURSES OPTION 1: Lesson menu opens lesson pages in new window. Learner manages window. Potential issue with pop-up blockers for some learners.

1. FOR MIGRATING EXISTING COURSES OPTION 2: Lesson menu navigates to (is replaced by) lesson window. Course lesson needs to manage returning to menu via URL.
2. FOR NEW COURSES: Overlay the lesson menu with the lesson window. Course manages DOM. Only asingle browser window, so no window management required.

Option 3 is preferred.

Sample Implementation of Course Navigation & Menu
The following is a reference implementation of decorating and adding listeners to a lesson list. It assumes that the lesson launch names and links are already written and queried (by either jquery, or *document.getElementsByClassName*).

The decoration of lesson links is done via css.

```
setupLessonLinks() {
  const context = this;

  let prevLessonStatus = null;

  let allComplete = true;

  this.getLessonLinks().forEach((/* Node */ el) => {

    const lessonLink = el.dataset.lessonLink;

    const lessonName = getLessonName(lessonLink); //potentially embed
in data-lesson-name?
    const lessonStatus = getLessonStatus(lessonName); //call to
localStorage

    //note we are using the SCORM lesson_status values

    allComplete &= lessonStatus === 'completed' || lessonStatus ===
'passed';
```

```
    //update icon; which css class to apply
    let linkClass = this.getLessonLinkClass(lessonStatus);

    // enforce pre-requisites; if the prev lesson not completed
    if (prevLessonStatus && prevLessonStatus != 'completed') {
      linkClass = 'lesson-link-locked';
    }

    //ie doesn't let us remove mult classes at a time, so we need to
hit each one individually
    el.classList.remove('lesson-link-check');
    el.classList.remove('lesson-link-in-progress');
    el.classList.remove( 'lesson-link-unlocked');
    el.classList.remove( 'lesson-link-locked');
    el.classList.remove( 'disabled');
    el.classList.add(linkClass);

    prevLessonStatus = lessonStatus;


    if (linkClass != 'lesson-link-locked') {
      const elClone = el.cloneNode(true);
      el.parentNode.replaceChild(elClone, el);
      //add handler to launch lesson
      elClone.addEventListener('click',
context.showLesson.bind(context), false);
    } else {
      el.classList.add('disabled');
    }

  });

  if(allComplete) {
    //show cert

  } else {
    //hide cert

  }

}
```

Certificate Generation

A Promise chain can implement the steps described in Section 1, but a sequence of functions andcallbacks will work just as well.

Sample Implementation for Certificate Generation

```
start() {

    // This dialog is simply a Bootstrap modal.
    this.createDialog()

        // resolves when learner enters name and clicks ‘Get Certificate’
        .then(this.collectLearnerName)

        // open another modal that shows iframe; pass info required to

        // render certificate via url to iframe
        .then(learnerName => this.showCertificate(learnerName))
        .catch(reason => {
            console.log(reason);
            $('.modal-backdrop, .certificateDlg').remove();
        });
}


createDialog() {
    return new Promise((resolve, reject) => {
        //use handlebars to render cert dlg
        const template =
require("templates/certificateDialog.handlebars");
        const resultHtml = template();
        $('.mainContainer .certificateDlg').remove();
        const jQueryContainer = $('<div
class="certificateDlg"></div>').appendTo('.mainContainer');
        jQueryContainer.html(resultHtml);

        resolve('dialog created');
    });
}

collectLearnerName() {
    return new Promise((resolve, reject) => {
        const dlgPtr = $('.certificateDlg > div:first-child');
        dlgPtr.modal().focus();
        dlgPtr.find('form').on('submit', function () {
            $('#certModal').off('hide.bs.modal');
            resolve($('input#name').val());
        });

        $('#certModal').on('hide.bs.modal', function () {
            reject('cancelled by learner');
        });
    });
}



rot13Fast(str) {
    const rot13Fast_input =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'.split('');
    const rot13Fast_output =
'NOPQRSTUVWXYZABCDEFGHIJKLMnopqrstuvwxyzabcdefghijklm'.split('');
    const rot13Fast_lookup = rot13Fast_input.reduce((m, k, i) =>
Object.assign(m, {[k]: rot13Fast_output[i]}), {});
    return str.split('').map(x => rot13Fast_lookup[x] || x).join('');
}


showIFrame(learnername, download)
    { if (download === undefined)
    {
        download = false;
    } else {
        //convert to true bool
        download = download == true;
    }

    const courseName =  getCourseTitle();
    const formatDate = function (datestring) {
        const dt = new Date(datestring);
        return `${dt.toLocaleString('default', { month: 'long' })}
${dt.getDate()}, ${dt.getFullYear()}`;
    }
    const formattedDate = formatDate(getPostAssessmentPassedDate());

    $('#preview-pane-wrapper').removeClass('d-none');
    $('#iframe-pdf').attr('src',
'../common/certificate/Certificate.html?' +
        `learnerName=${encodeURI(this.rot13Fast(learnername))}&` +
        `courseName=${encodeURI(this.rot13Fast(courseName))}&` +
        `formattedDate=${encodeURI(this.rot13Fast(formattedDate))}&` +
        `download=${encodeURI(download)}`
    );
}
```

PDF Generation

Sample Implementation for PDF Generation

The sample code below assumes:

1. String values are being passed on the URL query string
2. String values are obfuscated with rot13

```
function rot13(str) {
    const input =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
    const output =
'NOPQRSTUVWXYZABCDEFGHIJKLMnopqrstuvwxyzabcdefghijklm';
    const index = function (x) {
        return input.indexOf(x)
    };
    const translate = function (x) {
        return index(x) > -1 ? output[index(x)] : x
    };
    return str.split('').map(translate).join('');
}

function generatePdf() {
    const jsPDF = window.jspdf.jsPDF;

    const doc = new jsPDF({
        orientation: 'landscape',
        unit: 'in',
        format: 'letter',
        putOnlyUsedFonts: true
    });

    try {
        const hMargin = 0.5; //left/rt margin in in
        const vMargin = 0.25; //top/btm margin in in
        const width = doc.internal.pageSize.width;   // width of letter
in in
        const height = doc.internal.pageSize.height;   // ht of letter
in in
        const pageCenter = width / 2;

        //out background image has margins built-in
        doc.addImage("CMS_CertCompletion.png", "PNG", 0, 0, width,
height);
```

```
        //add learner name
        let lines = shrinkToFix(doc, learnerName, (width - hMargin *
4), 31, 30);
        let vOffset = lines.length === 1 ? 3.3 : 3.1;
        doc.text(lines, pageCenter, vOffset, {align: 'center'});

        //now do the course name
        lines = shrinkToFix(doc, courseName, (width - hMargin * 4),
26, 18);

        vOffset = lines.length === 1 ? 4.2 : 4;
        doc.text(lines, pageCenter, vOffset, {align: 'center'});

        //date
        doc.text(formattedDate, pageCenter, 5, {align: 'center'});
        if (download) {
            doc.save('certificate.pdf');
        }
    } catch (e) {
        console.error(e.message, e.stack, e);
    }

    if (typeof doc !== "undefined")
        try {
            if (
                navigator.appVersion.indexOf("MSIE") !== -1 ||
                navigator.appVersion.indexOf("Edge") !== -1 ||
                navigator.appVersion.indexOf("Trident") !== -1
            ) {

                $('#preview-pane')
                    .css('width', '100%')
                    .text('PDF preview is not supported by your
browser.  Click Download to retrieve and view your certificate.')
            } else {
                PDFObject.embed(doc.output("datauristring"),
"#preview-pane");
            }
        } catch (e) {
            alert("Error " + e);
        }
}


function shrinkToFix(doc, text, allottedWidth, singleLineMax,
doubleLineMax) {
    let fontSize = singleLineMax;
    let forceRepeat = false;
    let lines;
    do {
        forceRepeat = false;
        fontSize--;
        doc.setFontSize(fontSize);
        lines = doc.splitTextToSize(text, allottedWidth);
        if (lines.length > 1 && fontSize > doubleLineMax) {
            fontSize = doubleLineMax + 1;
            forceRepeat = true;
        }

    } while (lines.length > 2 || forceRepeat);

    if(hasRunt(lines)) {
        lines = fixRunt(lines);
    }

    return lines;
}

function hasRunt(lines) {
    //if only one line, nothing we can fix
    if(lines.length < 2) {
        return false;
    }

    const lastLine = lines[lines.length -1];
    const numWords = lastLine.trim().split(' ');
    return numWords.length === 1;
}

function fixRunt(lines) {
    let firstLine = lines[0];
    let lastLine = lines[lines.length -1];
    const firstSpaceAfterMidpoint = firstLine.indexOf(' ', Math.ceil(
firstLine.length / 2 ));
    const newFirstLine = firstLine.slice(0,
firstSpaceAfterMidpoint).trim();
    const toAppend = firstLine.slice(firstSpaceAfterMidpoint + 1,
firstLine.length).trim();
    firstLine = newFirstLine.split(' ');
    lastLine = toAppend.split(' ').concat(lastLine.split(' '));
    // const lastWord = firstLine.pop();
    // lastLine.unshift(lastWord);
    return [firstLine.join(' '), lastLine.join(' ')];
}

function getQueryStringVal(item) {
    const svalue = location.search.match(new RegExp('[\?\&]' + item +
'=([^\&]*)(\&?)', 'i'));
    return svalue ? svalue[1] : svalue;
}


const download = decodeURI(getQueryStringVal('download')) === 'true';
const learnerName =
rot13(decodeURI(getQueryStringVal('learnerName')));
const courseName = rot13(decodeURI(getQueryStringVal('courseName')));
const formattedDate =
rot13(decodeURI(getQueryStringVal('formattedDate')));

setTimeout( generatePdf, 0);
```

Tealium Code for Analytics

Contractors should add the Tealium code below that will allow the MLN to use Google Analytics to track their HTML scripted pages that aren't entered through Drupal.

Within the <head> of the page:
<script src="//tags.tiqcdn.com/utag/cmsgov/cms-www/prod/utag.sync.js"></script>

Immediately after the opening <body> tag of the page:
```
<script type="text/javascript">
  (function(t,e,a,l,i,u,m){
      t="cms-www";
      e=/^(www\.)?cms.gov/;
      a=(e).test(window.location.hostname)?
'prod':'dev';l='//tags.tiqcdn.com/utag/cmsgov/'+t+'/'+a+'/utag.js';i=document;u='script';m=i.
createElement(u);m.src=l;m.type='text/java'+u;m.async=true;l=i.getElementsByTagName(u)[0];l.parentNode.
insertBefore(m,l);
  })();
</script>
```

Section 3: Content Changes & Specification

To support the learner in the stand-alone environment (no LMS), we changed the standard wording used in the courseware.

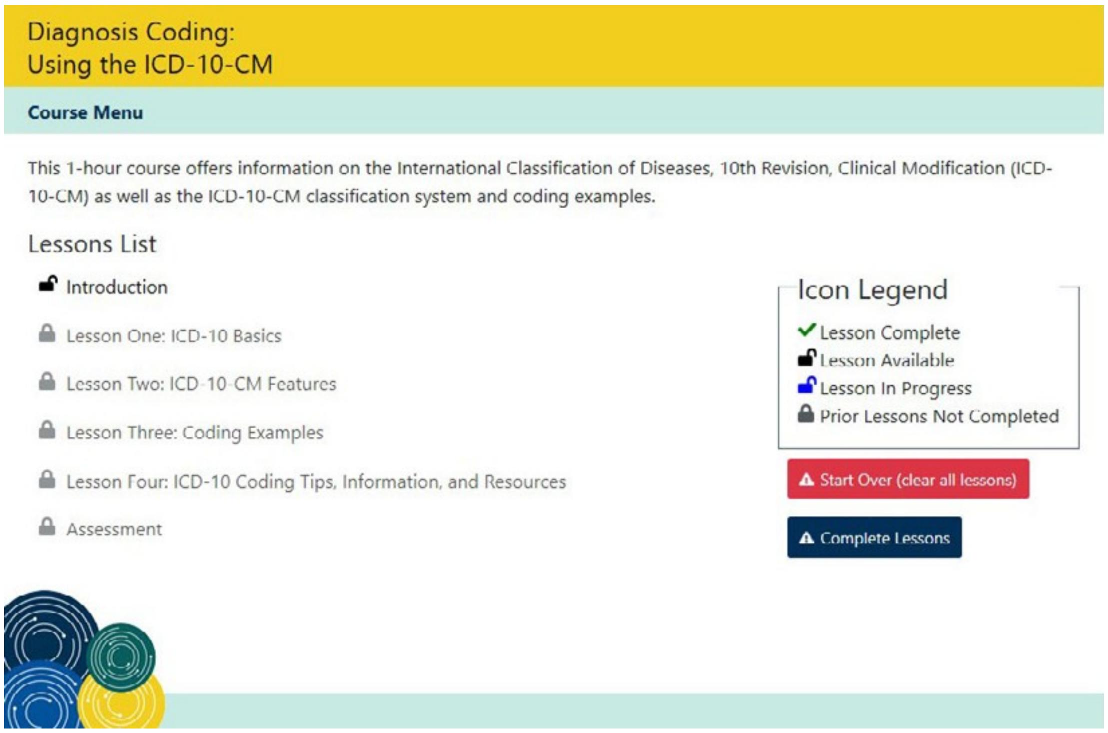
Menu Specification

The menu consists of 6 visual components:

1. Splash page (not really part of menu, but is shown prior to menu display)
2. Course synopsis (should contain estimated course duration and a brief description of course content)
3. Lesson list
4. Certificate button (available after passing assessment)
5. Legend (for any icons that need deciphering); denotes status of lessons- see wording in sample
6. “Start Over” button to clear progress status in all lessons

Optional: For testing on the contractor’s server, it’s useful to have a “Complete Lessons” button that completes all the lessons, allowing testers easier access to course lessons during testing. Important: make sure that for testing and production course builds provided to CMS, this button isn’t present.

Splash page
When you launch the course, you’ll see the splash page overlaying the course menu.
Clicking the “click to start course” button replaces the splash page with the course menu, which serves as dashboard.


Diagnosis Coding:
Using the ICD-10-CM

Course Menu

This 1-hour course offers information on the International Classification of Diseases, 10th Revision, Clinical Modification (ICD-10-CM) as well as the ICD-10-CM classification system and coding examples.

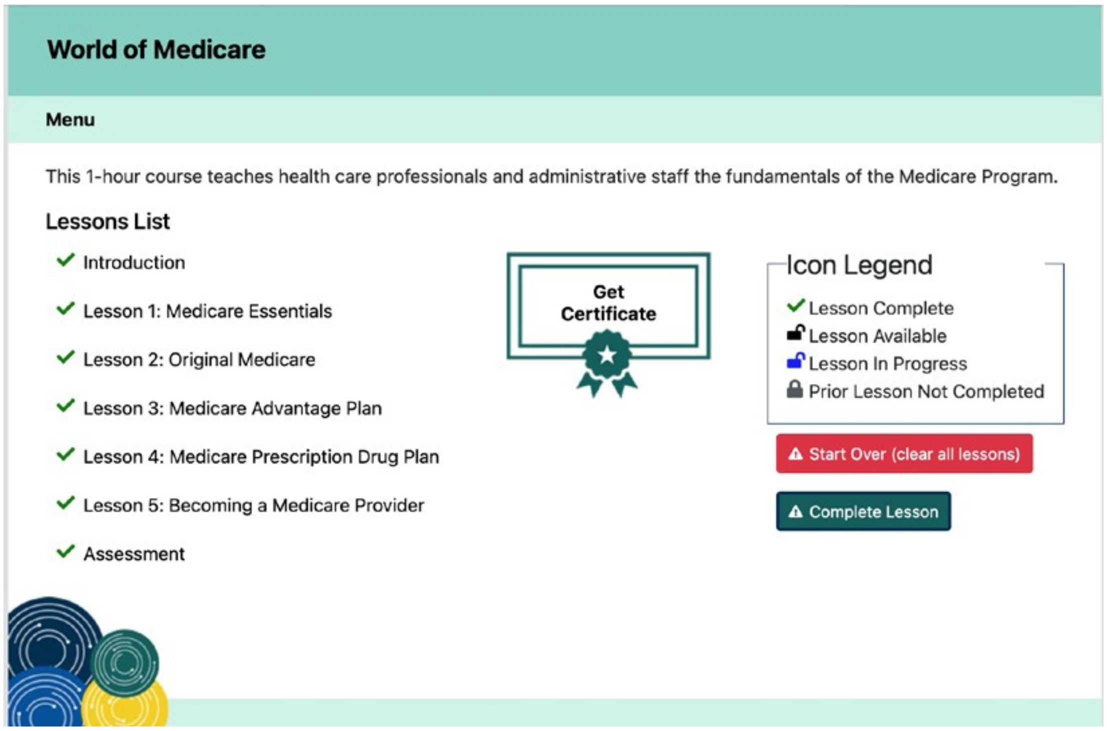
Lessons List

- Introduction
- Lesson One: ICD-10 Basics
- Lesson Two: ICD-10-CM Features
- Lesson Three: Coding Examples
- Lesson Four: ICD-10 Coding Tips, Information, and Resources
- Assessment

Icon Legend

- ✓ Lesson Complete
- Lesson Available
- Lesson In Progress
- Prior Lessons Not Completed

Figure 1. Sample of course menu prior to launching any lessons


World of Medicare

Menu

This 1-hour course teaches health care professionals and administrative staff the fundamentals of the Medicare Program.

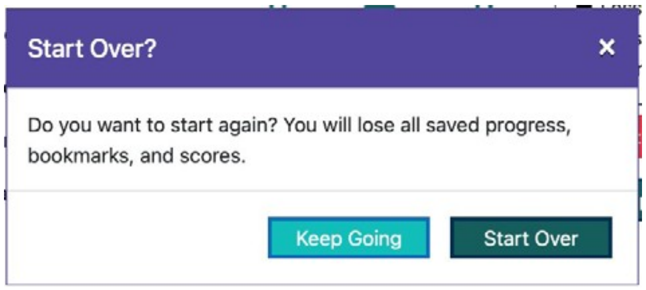
Lessons List

- ✓ Introduction
- ✓ Lesson 1: Medicare Essentials
- ✓ Lesson 2: Original Medicare
- ✓ Lesson 3: Medicare Advantage Plan
- ✓ Lesson 4: Medicare Prescription Drug Plan
- ✓ Lesson 5: Becoming a Medicare Provider
- ✓ Assessment

Figure 2. Course menu/lessons list page w ith Get Certificate button

Reset Course (Start Over button)

When you click the Start Over button, a confirmation dialog is shown explaining the effects if continuing the procedure. Use the following instructional text.


Start Over?

Do you want to start again? You will lose all saved progress, bookmarks, and scores.

Figure 3. Start Over modal dialog

Course Evaluation

We removed all references to the Course Evaluation in the introduction, post-assessment, and help. Use the following language to describe the steps entailed to earn a certificate.

To get your certificate, you must successfully complete all lessons, in order, and pass the assessment with a score of <passing score>% or higher.

Exit button renamed

The Exit button has been renamed Course Menu. This will require several changes

- Change button label in header on all pages
- Change instructional text on Assessment Result – Fail page
- Change instructions and toolbar screenshot in help

Post-Assessment renamed to Assessment

We retitled the Post-Assessment to Assessment. Use this on all pages that refer to the Post-Assessment. Potential locations include:

- Menu – Assessment lesson title
- Help – lesson listing
- Help – Post Assessment section
- Intro – how to obtain certificate
- Intro – this course consists of…
- Intro – lesson listing
- Last content lesson – …click Continue to return to the menu, and then select Assessment
- Assessment intro
- Assessment summary

Menu Button renamed

The button at the end of each lesson (that returns learner to course menu) is now called the Continue button.

Use the following language at the end of each lesson:

> Click the Continue button below to return to the course menu. Then, select <next lesson title>.

Wording Changes to Introduction

We reworded the introduction to remove references to the LMS, evaluation, and revise the certificate instructions.

Intro – MLN Page
The link to the LMS was removed:

About the Medicare Learning Network®

Medicare Learning Network®

The Medicare Learning Network® (MLN) offers free educational materials for health care professionals on CMS programs, policies, and initiatives. Get quick access to the information you need.

- MLN Publications & Multimedia
- MLN Events & Training
- MLN News & Updates

Intro – Course Content

We removed the references to Evaluation and revised the Certificate Instructions:

Course Content
This course consists of reference documents, course content, review questions, and an assessment. Successful completion of this course requires a score of 70% or higher on the assessment.

This course uses cues at various times to give additional information. The cues are hyperlinks, buttons, rollovers, and pop-up windows. For more information on using these cues, click the Help button in the top right corner. The Reference button includes resource documents and a glossary of terms defined within this course. You may print these materials any time during this course.

After you successfully complete the course, you'll get instructions on how to get your certificate.

Wording changes to Assessment:

- Reworded and removed references to LMS, evaluation, and revised certificate instructions
- Changed from post-assessment to assessment

Assessment – Intro
We modified the certificate instructions. Use this new language for requirement for course completion.

Assessment
Let's see how much you have learned. This assessment will ask you to answer 10 questions related to Medicare. The estimated completion time for this assessment is 15 minutes.

You'll be able to change your answer until you select the Done button. Once you click on Done you will not be able to change your answer. After clicking on Done and reviewing the feedback to your answer, click Next to continue to the next page. Once you click Next, you will not be able to exit and save your progress.

Upon successful completion of this course, you'll get instructions on how to get your certificate. Successful completion of this course includes completion of all lessons and a passing grade of at least 70% on the assessment.

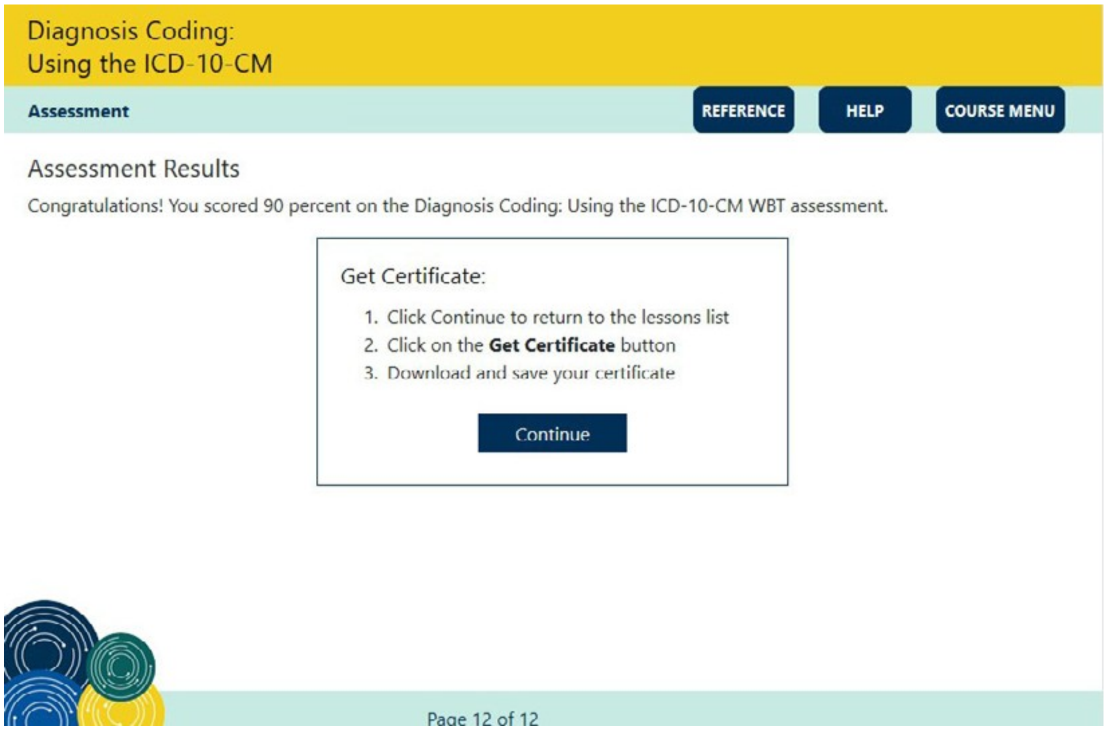
Assessment Results – Passed
We revised the certificate instructions and removed the print button.

Assessment Results
Congratulations! You scored 80 percent on the World of Medicare WBT assessment.

Get Certificate:

1, Click Continue to return to the lessons list
2. Click on the Get Certificate button
3. Download and save your certificate

[Continue]


Diagnosis Coding:
Using the ICD-10-CM

Assessment Results

Congratulations! You scored 90 percent on the Diagnosis Coding: Using the ICD-10-CM WBT assessment.

Get Certificate:

1. Click Continue to return to the lessons list
2. Click on the **Get Certificate** button
3. Download and save your certificate

Figure 4. Assessment passed page in assessment w ith certificate instructions- note that the Continue button returns the learner to the course menu and lessons list. The certificate can't be accessed from w ithin a lesson.

Assessment Results
Good try! You scored 50 percent on the World of Medicare WBT assessment.

However, you did not successfully pass the course. Please retake the assessment. Click the Course Menu button to return to the lessons list and take the assessment again.

Click Print to print this page.

[Print]

This page refers to the Course Menu button instead of the Exit button.

Help Content

The help content has been significantly reworked and standardized with a few sections that need adapting for each course. We standardized the remainder of the text. Don't change this language on a per-course basis. Use the standardized text.

1. Estimated duration of course
2. Lesson list
3. Definition rollover description (color and/or decoration)
4. Passing score of 70%)

Refer to all course units as Lessons. All other designations (for example, Modules) are no longer permitted.

Use the following copy as the template for help pages. Adapt the highlighted sections to the course specifics.

Welcome to the Help Page

Course menu

Course cues

How to take this course

Get Certificate

Questions, Comments, & Help

About training: Email us at MLN@cms.hhs.gov

About CMS.gov: Read our Web Policies & Important Links

This web-based training (WBT) course consists of course lessons and a knowledge check. The course should take approximately 1 hour to complete:

Course Menu

Introduction

Lesson Name

Lesson Name

Lesson Name

Lesson Name

Assessment

To get your certificate, you must successfully complete all lessons, in order, and pass the assessment with a score of 70% or higher.

TOP

Course Cues

The course uses cues to provide more information. The cues are hyperlinks, buttons, rollovers, and pop-up windows.

Hyperlink
An underlined word or graphic on the page that takes you to a webpage. When clicked, it opens the webpage in a new window. To return to the course, close the window. Hyperlinks appear in blue underlined text.

Button
A figure or button that the learner clicks to select an option.

Rollover
When you roll your mouse over a word or acronym it's underlined with a dashed, dark-red line, and an explanation of the word or phrase will appear on the screen.

Pop-up Window
A window that opens (pops up) when you select certain option.

TOP

How to Navigate This Course

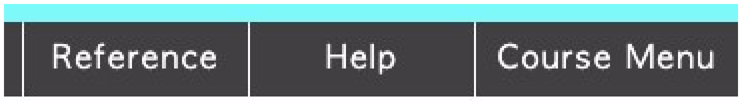
Next and Back buttons appear on pages in the lessons

Click Next to go to the next page in the lesson

Click Back to go to the previous page in the lesson

Print a page by pressing **CTRL + P** or the print button

Click the continue button to return to the Course Menu to access lessons and the assessment.



The toolbar contains 3 buttons that you may select at any time.

1. Reference—Access course resources
2. Help—Get course help
3. Course Menu—Return to the Course Menu with lessons list

By clicking the Start Over button, you'll start over and lose all saved progress, bookmarks, scores, and certificates, and you won't be able to get a new certificate until you successfully complete the course again.

TOP

Get Certificate
Pass the assessment with a 70% or higher

Click Course Menu to return to the lessons list

On the lessons list page, click on the **Get Certificate** button

Enter your name as it should appear on the certificate

Click on the **Get Certificate** button to preview the certificate

Click on the **Download** button and save your certificate

Note: If you don't successfully pass the assessment with a 70% or higher, select Course Menu and go back to the assessment to take it again.There's no limit to how many times you can review the lessons or take the assessment.

TOP

Certificate Learner Flow

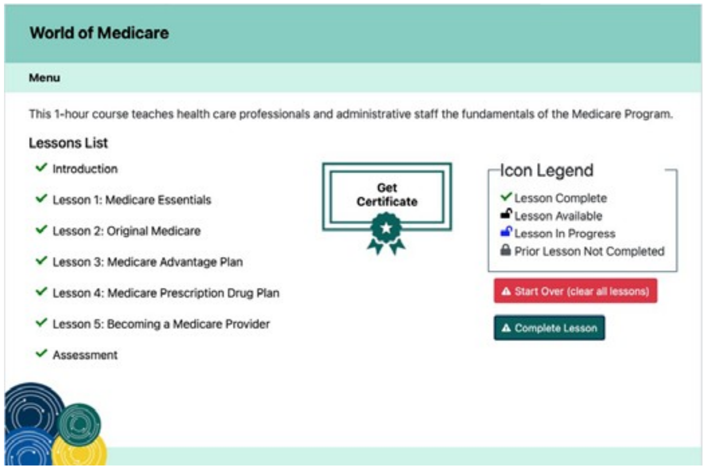
The course generates the certificate. The Get Certificate instructions tell learners to save. Learners keep their own records in absence of a LMS transcript.

- See Help Instructions
- See also Pass or Fail pages for assessment
  - Removal of administrator signature; use Rich Cuchna signature
  - Date on certificate is the date the assessment was passed, not the current date. Use the dateformat <full month name> <date>, <yyyy>.

Learner flow is as follows:

Step 1: Show Certificate button

When learner successfully completes (passes) the assessment, the certificate button is shown on the Course Menu page. The label is, Get Certificate.



World of Medicare

Menu

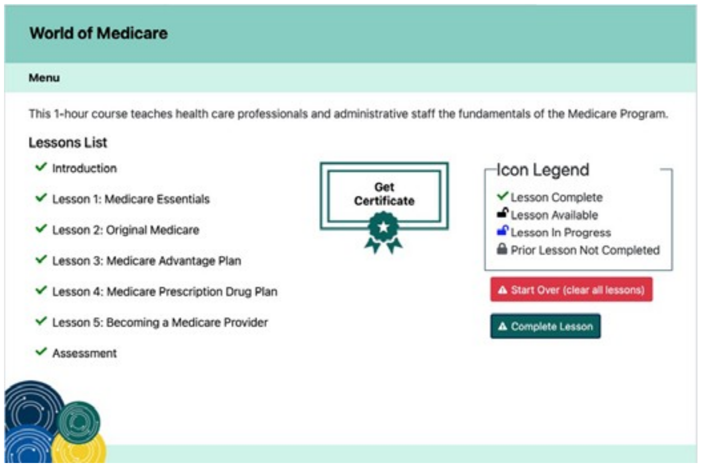
This 1-hour course teaches health care professionals and administrative staff the fundamentals of the Medicare Program.

Lessons List

- ✓ Introduction
- ✓ Lesson 1: Medicare Essentials
- ✓ Lesson 2: Original Medicare
- ✓ Lesson 3: Medicare Advantage Plan
- ✓ Lesson 4: Medicare Prescription Drug Plan
- ✓ Lesson 5: Becoming a Medicare Provider
- ✓ Assessment

Step 2: Certificate instructions and Learner name field

When learner clicks certificate button, dialog appears with certificate instructions and input box for learner's name.Use the following Wording for instructions:



World of Medicare

Menu

This 1-hour course teaches health care professionals and administrative staff the fundamentals of the Medicare Program.

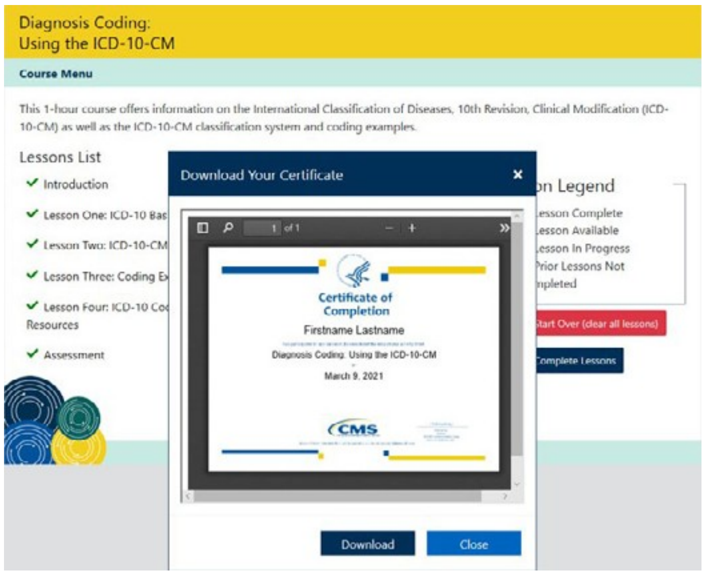
Lessons List

- ✓ Introduction
- ✓ Lesson 1: Medicare Essentials
- ✓ Lesson 2: Original Medicare
- ✓ Lesson 3: Medicare Advantage Plan
- ✓ Lesson 4: Medicare Prescription Drug Plan
- ✓ Lesson 5: Becoming a Medicare Provider
- ✓ Assessment

1. Enter your name as you would like it to appear on your certificate
2. Click Get Certificate
3. Choose a location to save your certificate

The button labels are:

- Get Certificate
- Cancel



Diagnosis Coding: Using the ICD-10-CM

Course Menu

This 1-hour course offers information on the International Classification of Diseases, 10th Revision, Clinical Modification (ICD-10-CM) as well as the ICD-10-CM classification system and coding examples.

Lessons List

- ✓ Introduction
- ✓ Lesson One: ICD-10 Bas
- ✓ Lesson Two: ICD-10-CM
- ✓ Lesson Three: Coding Ex
- ✓ Lesson Four: ICD-10 Co

Resources

- ✓ Assessment

Step 3: Preview Certificate
When learner enters their name and clicks Get Certificate,a preview of the certificate appears. The label Get Certificate changes to Download, which, when clicked, downloads the PDF.

Step 4: Download Certificate
The learner can download the certificate after previewing the title.

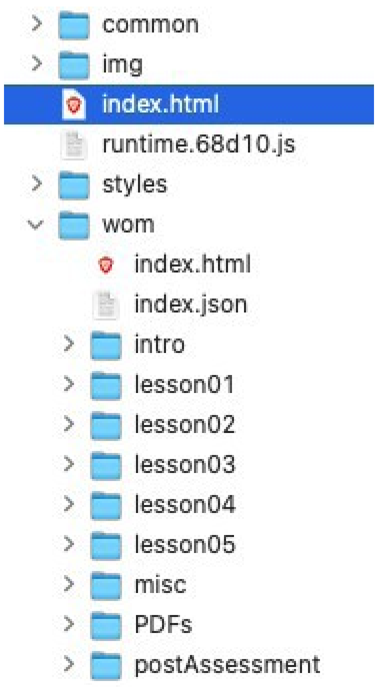
Section 4: Packaging Requirements Delivery of Courses:

Deliver the course as a zip file. The final zipped course files are available for download from the contractor site.

The zip file shouldn't contain the .xsd files, or the .xml files used by a SCORM LMS, unless they're required to generate the course menu.

File Structure

The structure of the course files hasn't changed much from the structure used by the LMS. However, there's no discovery mechanism for the course menu. This functionality needs to be added (and discovery is performed by convention instead of configuration).



```
> common
> img
  index.html
  runtime.68d10.js
> styles
v wom
    index.html
    index.json
  > intro
  > lesson01
  > lesson02
  > lesson03
  > lesson04
  > lesson05
  > misc
  > PDFs
  > postAssessment
```

Discovery Convention
Use the following convention:
In the root of the zip file, there must be an index.html file. This file can either:

1. Implement the course menu, or
2. Implement a redirect (JavaScript redirect, or meta-refresh) to the course menu. Note that this redirect must be relative.

Instructions for File Load to CMS

The website administrator will unzip the course into the folder on the webserver and will assume that the index.html file in the root will launch the course.

Instructions for URL

Refer to the Naming and Place a Product section for instructions on WBT URLs.