Updating a Legacy Courseware Viewer

Impression Scribe migration guide

Use this guide when moving an existing Impression CCT project to Impression Scribe while retaining its HTML5 JavaScript Courseware Viewer. Update the Viewer to BCF 2.0.0 and install the included canvas compatibility extension where needed.

Converting a project updates its database and authoring settings. It does not automatically update a separate or customized Viewer. The canvas extension keeps the BCF source file unchanged.

Before you begin

- Back up the project database and the complete Viewer folder. Work on a copy until preview and published courseware have been checked.
- Convert the project with the current Scribe converter. Review the converted storyboard templates in Project Settings.
- Have bcf-2.0.0.js available from your Impression framework distribution. This package contains the guide and bcf-2-legacyCanvas.js; it does not include BCF itself.

Viewers are framework implementations and may differ in filenames, folder structure, and custom behavior. The paths below use the older standard Viewer. Retain your existing customizations and adapt the equivalent locations in your implementation.

Update the BCF reference

Place bcf-2.0.0.js in assets/source and update the active script loader to load it instead of the older BCF file. In the standard source Viewer, the loader is assets/source/rte.html. Load only one BCF version.

```
<script src="assets/source/bcf-2.0.0.js"></script>
```

Update the sequencer constructors

In assets/source/viewers, make these constructor substitutions. Keep the surrounding Viewer logic, settings, and event subscriptions.

CommandSequencerViewer.js

```
new bcf.CommandSequencer()
    becomes new bcf.SequencedInteractionPlayer()
```

TimeSequencerViewer.js and AudioSequencerViewer.js

```
new bcf.TimeSequencer()
    becomes new bcf.SequencedPresentationPlayer()
```

Install the legacy canvas extension

Use this extension with an older CanvasViewer that expects the former built-in canvas actions. It restores recognition of those action names and adapts popup text and graphic reads to converted data. A current Viewer that already implements custom actions does not need it.

1. Copy bcf-2-legacyCanvas.js from this package into assets/source alongside BCF.

2. In the active loader, retain the BCF reference and the existing CanvasViewer.js reference. Add the extension and installation call after CanvasViewer.js. Keep all other Viewer script references.

```
<script src="assets/source/bcf-2.0.0.js"></script>
<!-- Keep the other existing Viewer scripts. -->
<script src="assets/source/viewers/CanvasViewer.js"></script>
<script src="assets/source/bcf-2-legacyCanvas.js"></script>
<script>bcf.legacyCanvas.install(CanvasViewer);</script>
```

3. Reload or restart the Viewer before testing. Installation must run after BCF and CanvasViewer are defined, and before any storyboard is loaded. The final line is inline JavaScript that activates compatibility; loading the extension alone does not enable it.

What the extension supports

- The historical jump, popup text, popup graphic, play sound, hyperlink, command, and display range action names. Legacy case variants such as PopupText and popupText are recognized when no exact custom registration takes precedence.
- Both the original flat popup properties and the newer nested popup text and graphic properties, through the existing legacy popup handlers.
- Unrelated custom action registrations remain available. Conflicting definitions of the historical constants produce an explicit error rather than being overwritten.

The extension is for BCF 2.0.0. It wraps handlers named onAction_popupText and onAction_popupGraphic. If your Viewer uses different handler names or reads popup data asynchronously, its developer must adapt the integration. It does not implement arbitrary custom actions.

Only popup text and graphic payloads are adapted. Recognition of the other historical action names does not by itself adapt changed navigation, sound, hyperlink, or command payloads. Test each action used by your project.

If your Viewer uses a compiled runtime

The standard Viewer tries assets/rte.min.js first and uses rte.html only when that bundle is unavailable. If a compiled bundle is present, editing rte.html alone will not update the running courseware.

Rebuild the bundle with one BCF version, then the Viewer code, the extension, and the installation call in that order. An inline call in rte.html is not automatically included in a JavaScript-only build. Put the call in the compiled startup code after CanvasViewer is defined and before storyboard loading. Include the extension only once.

Test the updated Viewer

Reload the preview after changing scripts. Then create a fresh courseware export from the converted project and test that package separately. Existing exported lesson XML does not update when the source database changes.

- Open a normal content storyboard and confirm that text, graphics, and navigation behave as expected.
- Test Command, Time, and Audio Sequencer storyboards used by the project. Check rendering, cue timing, play, pause, restart, and completion as applicable.
- Select canvas popup text and graphic elements. Check their content, closing behavior, interaction styling, and keyboard access.
- Test every other canvas action used by the project, particularly jumps and display ranges. Confirm the destination and completion behavior.
- Navigate away from and back to the storyboards. Check completion tracking and, where applicable, the LMS resume behavior.
- Test the published output in the browsers and delivery environment used by your learners.

The supplied extension was exercised with a legacy sample project. Older and customized Viewers still require their own acceptance testing.

If something does not work

The Viewer reports that a sequencer is not a constructor

Confirm the constructor substitutions on page 1 and check which BCF script is actually loaded. A cached script or an older compiled bundle can hide source changes.

Canvas elements do not respond

Check that the extension loads successfully and that its installation call runs before storyboard loading. Inspect browser errors for a BCF version or action-registration conflict. Confirm that your Viewer uses the expected legacy handlers.

A popup opens with no content or unexpected content

Inspect the exported action data and any customized popup handler. Converted popup payloads are nested. The old renderer may use the element name as its heading; the extension does not add support for a separate custom title field.

The updated source works but the export does not

Check the export template or packaging process. Include the updated Viewer scripts and any rebuilt runtime, and regenerate lesson data from the converted project.

Return to the previous Viewer

Restore the backed-up Viewer folder and its matching project/export data. To remove only the canvas extension from a Viewer that can operate without it, remove its script reference and installation call, then reload. Removing compatibility does not reverse database conversion.

For help with a customized Viewer, contact Logicdriven support with the BCF version, Viewer files, affected storyboard data, and the browser error message.