# Live Edit

Live Edit (formerly known as "Edit on the Fly" allows a teacher/researcher to modify a sequence while is running. This happens in Monitor when a lesson is running in the Sequence tab.

In the sequence tab we have a button that says "Live Edit". If pressed, it pops up a message saying that a permission gate will be added to the next transition of the learner that is the furthest into the lesson (if at least one learner has finished, a message pops up saying that this lesson cannot longer be edited). This running sequence now is flagged as being edited, and no other teacher/researcher can edit it again (unless is the same user that set this flag before).

The Authoring interface will load directly in the Monitor window replacing it completely.

The edit on fly version of authoring will have a restricted toolbar - it will need Apply Changes, Cancel Changes, Copy, Paste, Transition, Optional, Flow and Group. There's no save option as teachers should only Apply Changes whenever they are completed all modifications

The menus will also need to be changed to reflect the monitoring functionality.

- File: Apply Changes, Cancel Changes.
- Edit: as normal
- Tools: as normal
- Help: as normal

Cancel changes will close the authoring window without saving changes and will resume the lesson (ie clear the edit on fly flags and the system gates).

While Edit-on-the-Fly, the teacher can edit the remaining "unfinished" activities in the lesson. All previous activities are locked (and a padlock like icon is drawn above them?). The teacher can now add/remove the activities.

A System gates gets added right after the last student completed activity. This System gate is to be removed once the Apply Changed button is pressed.

When the teacher clicks on Apply Changes (either on the menu or on the toolbar), the design will be saved in the database. We will have a custom dialog box that says that Changes have been applied to the lesson and a "Done" button. After the user presses the Done button, then the authoring screen reloads the Monitor interface on the Sequence tag.

Note that at this stage **non** System gates should appear in the sequence/lesson.

If the teacher closes the edit on fly / monitoring windows (or the browser crashes) before they complete editing, then when that teacher reopens monitoring, the teacher will be taken to the sequence tab and the edit on fly window launched automatically.

If another teacher goes into monitoring while the design is being edited, then they will see a message on the lesson tab indicating that the lesson is being modified and by whom it is being modified.

The edit on the fly lock is only released when the editing teacher clicks "Apply Changes".

**New/Changed database fields:**

Activity:

- boolean readOnly. This indicates that the activity has been commenced and cannot be edited.

Learning Design:

- boolean editOverrideLock. New field. Used to override the normal readOnly setting for edit on fly.
- Long editOverrideUser. New field. Indicates the user who is editing the design via edit on fly. Will be passed to Flash as the user id but does not need to be checked by Flash.
- Integer designVersion. New field. Used by learner to detect that the design has changed and hence the progress bar will need to be updated. It will have the side affect of ensuring that two people editing the same design in authoring will not cause the one person's changes to be lost - as the second person's changes should be rejected.

| readOnly | editOverrideLock | Description |
|---|---|---|
| false | true or false | Not locked. May be edited. |
| true | false | Locked. May not be edited by anyone. |
| true | true | Locked for edit on fly. May be edited by the user indicated in the editOverrideUser field. |

On the Flash end, the client should check that the design is either readOnly = 0 or readOnly = 2. It should not allow anyway to edit a design that is readOnly = 1.

Lesson:

- Hibernate synchronisation field. New field. Used to ensure synchronisation across clustering. Will not be passed to Flash as is of no interest to Flash
- boolean lockedForEdit. New field. Used to stop the learner's progressing while the stop gates are being configured. Released once stop gates are created and hence is different to the Learning Design's editOverrideLock logic.

On the server end, the save validation will check that the design either readOnly = 0 or readOnly = 2. If it is readOnly = 2, then it will check that the user saving the design is the user specified in editOnFlyUser.

The new permission gate that is added when the edit on the fly is selected will be marked as locked (don't want the teacher removing our "stop the users" point!). The java code that creates the permission
gate will need to assign it x/y co-ords, probably half way between the previous and next activities. The current transition will need to be remove and replaced with two new transitions.

### Stopping the Learners from Proceeding Using Stop Gates

Initially we just need one permission gate, but once we do branching we will probably need more than one stop gate (for if learners are on branches). We will create a new type of gate, SystemGate, which is a subclass of PermissionGate. For the moment, it won't have any new features but it allows us to distinguish between user created permission gates and our system generated gates.

We can't just instantly create the gates, so setting up the gates will be a multi-step process. When a staff member clicks edit:

```
If design is already being edited by another user then
    return a message to the user stating that it is being edited and by whom.

Transaction 1 starts.
    If the design isn't being already set to being edited then
       set the lesson to locked and set the learning design set to edit.
       The lesson will include a Hibernate version column, so if
       two people try to set the lock flag at the same time, the second
       persons's transaction will fail.
Transation 1 ends.

Transaction 2
     Go through the learning design and based on the locked activities, set new stop gates.
     Mark lesson as unlocked.
Transaction 2 ends.

Editing screen is then displayed to the user.
```

I suspect the web calls for the monitoring calls will have to NOT use the open session in view filter, so that the database transactions can be completed and hence have a valid lock for other machines in the cluster. If that is not possible, then it will need to be done as two different web calls.

To reduce the chances of collision between the lock being set, and to reduce the chances that a problem with generating the next activity will cause a tool to rollback, the current completeToolSession is split into two different calls. CompleteToolSession will now only mark an activity as completed, it won't calculate the next activity. Instead of forwarding to a screen that calls an already calculated next activity, the tool will forward to a screen that requests that the server calculate the next activity.

```
Learner hits finished button in tool.

Transaction starts:
    Tool records user finished within tool.
    Tool calls complete Tool Session
        Complete Tool Session does:
            Marks activity(ies) as complete.
            Returns "get next activity URL"
Transaction ends.
Tool forwards to the URL.

Get next activity screen calls the server to go to the next activity.
Transaction starts
    If ( Lesson locked)
        URL = "Lesson under edit" screen.
    Else
        Works out the next activity.
        Marks the next activity as locked.
        URL = URL of the next activity
Transaction ends.
Forwards to the  URL.
```

This change to the progress engine means that there isn't a guaranteed current activity (ie if the complete tool session runs when the lesson is locked). So join lesson, resume lesson, etc will need to be changed to cope with this.

When the staff member has finished their changes and saved a valid design, all the system stop gates are removed. This requires that they are removed from the design, and if they appear in any learner's progress details then they must be removed from the progress details (or we get foreign key exceptions). If the learner gets as far as the stop gate, then the stop gate will be the learner's current activity and in the their attempted activity lists. So we check all the progress details and remove any entries for the removed gates, just in case.

If the progress engine is not able to calculate a user's next activity (because an activity has disappeared) then it will go back to the beginning of a design and calculate where the user should be. So when the stop gates are removed, the next time the learner who is on one of the stop gates tries to advance (either by rejoining or the gate page refreshing) then the progress engine will detect the lack of a current activity will recalculate an appropriate current activity. This logic will also be useful if the design become corrupted for another reason.

Now, it will be possible for a learner to "escape" the lock, and get passed the stop gates. In a clustered environment, if we are really unlucky then the code to mark the lesson as locked could occur simultaneously with a learner getting their next activity. If this occurs, the learner will avoid the stop gates and will continue with the lesson. To stop this occurring, we would need to put a lock on the lesson table and this would potentially create a bottleneck in the progress engine - and that's the last thing we want! But allowing the learner to continue would wreak havoc as they would lock activities as they go, and the staff member wouldn't be able to change the sequence in authoring. So the compromise is to not put a lock on the lesson table but to check the activity status on each step through the design. Now when the progress engine goes to advance to the next activity it checks if it is currently in "Live Edit" mode and if it is that the activity is already read only. If the lesson is not in live edit mode, or it is and the activity is read only, then the learner can do the activity. If not, then they will go to an error screen. If this

When the editing is finished, the current learners' progress bars need to be updated. Therefore the "next activity" page will pass to Flash both the updated progress and a version number for the learning design. When the version number changes, the Flash module will request a fresh copy of the learning design.

When we get an instant messaging service, it would be nice to send a message to all learners in the lesson that the lesson has been changed, and telling them when the progress bar will be updated.

### Deleting Activities and Stop Gates

If a learner has not commenced an activity, then the activity can be deleted. This has the following implications for other tables in the database:

- When a lesson is started, the tool sessions for all non-grouped activities are created. So, when an activity is deleted, authoring now checks to see if there is a related tool session and if there is, then it deletes the tool session. If it finds more than one tool session then it throws an exception as this is an unexpected case.
- When the tool sessions are created, the tool's also set up their own tool session records. These become orphan records (along with their matching tool content records) when an activity is deleted.

System activities such as gates and grouping do not have a tool session, so their tool sessions do not need to be removed.

One day it would be nice to have a clean up job that deletes orphan content and session records in the tools.

### Saving an Invalid Design

In LAMS 2.0, it is possible to save an invalid design. The normal way this occurs is if transitions are missing. All other errors (as of Live Edit being implemented) would be due to a bug in the Flash interface.

However we do not want an invalid design saved if possible. For example, if the design is saved with a transition missing, the server may calculate the wrong activity as the first activity and any new learners might get an activity PASSED the stop gate as the first activity. They will also get progress bar that has only part of the design. If this occurs it won't do any damage - the live edit + read only check mentioned previously will ensure that the lesson is not damages. However it will look odd and is likely to confuse people.

It is not easy change the server to not save an invalid design (as it would mean a major rewrite of the design update code or relying on a database rollback)  so the validation to check for missing transitions will be added to the authoring client for edit on the fly. If the staff try to save and exit the live edit mode (using the Apply Changes button), the client will detect the invalid transition and not let them proceed.

However if the staff does an insert/merge then the design will become invalid. (Insert/merge was introduced in LAMS 2.1.) When they do the insert, it will save the current design (irrespective of the validity) and call the insert method on the server. Even if the design wasn't invalid before, it will be after the insert. This isn't the most optimal situation, but it is impossible to avoid given the current implementation of insert/merge. Cancelling is now not possible so the Cancel button is disabled. If the staff try to save and exit the live edit mode (using the Apply Changes button), the client will detect the invalid transition and not let them proceed.

### Converting Data From LAMS 2.0

An update script alter_21_editonfly has been written to add the new fields to an existing LAMS 2.0 database. This script also sets up the values for existing rows in the lams_lesson, lams_learning_design and lams_activity tables. The update will lock (set to read only) ALL activities that are attached to a preview lesson or a normal lesson (ie if the learning design has copy_type_id = 2 or 3). This will ensure that all activities that have been attempted by learners will be locked.

As a result, the Edit On Fly functionality will not be useful for any existing lessons - staff will only be able to add activities to the end of existing lessons.  The Edit On Fly functionality will become useful once new lessons are created on the upgraded server.

Locking all activities for existing lessons is much simpler than trying to work out which activities have actually been accessed, and in keeping the code simpler we reduce the risk of an error existing in the data conversion code.

### Export Portfolio

Export Portfolio is not affected by Live Edit.

In the learner version of export portfolio, the activities exported are based on the learner's completed and current activities. These activities will all be read only, and cannot be deleted. So the export portfolio should function as normal.

The the monitoring mode (class export portfolio) would be affected by invalid designs, but it isn't accessible during  live edit - the editing staff member will be on the edit screen and the other staff members can't get to the sequence tab.