

2023-04-13 DSpace 7 Working Group Meeting

- [Date](#)
- [Agenda](#)
- [Attendees](#)
- [Current Work](#)
 - [Project Board](#)
 - [New Feature development process for 7.6](#)
 - [Issue Triage process for 7.6](#)
- [Notes](#)

Date


13 Apr 2023 from [14:00-15:00 UTC](#)

Location: <https://lyrasis.zoom.us/my/dspace?pwd=RTk4QUhISnhPRi9YenVrTFJKbDIIQT09> (Meeting ID: 502 527 3040). **Passcode:** dspace

- More connection options available at [DSpace Meeting Room](#)

7.6 Release Plan (Tentative)

 Release Schedule:

-  **Friday, April 7 (Donated Feature Notification Deadline):** Any community members who wish to donate a feature to this release must notify [Tim Donohue](#) by this date (either via email, Slack or GitHub). The DSpace 7 team will then provide feedback on whether it will be possible to include this feature in the release (based on team member availability and the size of the feature). Early notifications are more likely to get included in the release.
- **Friday, Apr 28 (Feature PR Creation Deadline):** PRs should be created by this date if they are to be reviewed in time for the release. Please note there is no guarantee that a PR will be included if it is created by this date. Larger PRs are recommended to be created earlier, as that makes it more likely they can be reviewed in time for inclusion. *(Smaller bug fixes are welcome anytime)*
- **Friday, May 12 (Feature PR Review/Test Deadline):** All code reviewers or testers should submit their feedback by this date. Code reviews must be constructive in nature, with resolution suggestions. Any code reviews submitted AFTER this date will be considered non-blocking reviews. *NO TE: Larger PRs or donated PRs may have their own deadlines established for PR creation, review and merger.* This deadline only applies for PRs with no other deadline established.
- **Friday, May 19 (Bug PR Creation Deadline):** Bug fix PRs are still acceptable after this date if they are very high priority. However, any submitted after this date will likely need to have pre-assigned reviewers in order to ensure the review can be completed before the PR Merge Deadline.
- **Friday May 26 (PR Merge Deadline):** All PRs should be merged by this date. (Note: bug fixes can still get in after this deadline, as long as they are small or important)
- **Week of May 29-June 2 (Documentation & Release Week):** Any merged PRs which don't have minimal documentation (how to enable / configure) MUST have documentation created this week. Later in this week (around Thurs) will be the release.
- **Monday, June 5:** Public Release Announcement. 7.6 will be announced/released by this date.

Ongoing/Weekly tasks:

- Tackle/Claim issues on [7.6 board](#) (starting with "high priority")
- Review/test all PRs assigned to you for review/testing: <https://github.com/pulls/review-requested> (Prioritize reviews of "high priority" PRs first)

Agenda

- (30 mins) General Discussion Topics
 1. DSpace [7.6](#) ongoing development
 - a. Very High priority bugs located in 7.5:
 - i. Submission form "Type" dropdown changes values when pressing Enter: <https://github.com/DSpace/dspace-angular/issues/2145> Might be related to Safari browser dropdown issues: <https://github.com/DSpace/dspace-angular/issues/2124>
 - b. 7.6 code review process brainstorms: See [Incentivizing Code Reviews and PR Testing](#)
 - c. *Releases after 7.6:* Later releases will be bug-fix only.
 - i. Tim will bring to Steering the suggestion to switch post-7.6 release numbering to 7.6.1, 7.6.2, 7.6.3 (for eventual bug fix release). This clarifies that 7.6 is the final feature release, and that every later release is a minor upgrade.
 2. Pierre Lasou and Jeff Morin from U of Laval will join to discuss their work on porting REST API Reporting capabilities to v7
 - a. Ticket: <https://github.com/DSpace/DSpace/issues/7641>
 - b. Frontend PR: <https://github.com/DSpace/dspace-angular/pull/2163>
 - c. Backend PR: <https://github.com/DSpace/DSpace/pull/8598> and Contract PR: <https://github.com/DSpace/RestContract/pull/202>
 3. *(No Major Updates this week)* Demo Site maintenance (<https://demo7.dspace.org/> and <https://api7.dspace.org/server/>)?
 - a. LYRISIS is still working on this, but some restructuring of plans has had to occur because of internal deadline changes. Likely not to be completed until late April.
 - b. *In the meantime*, can we ensure that it is still possible to send updates to both the frontend & backend per instructions at [Updating DSpace 7 Demo Sites](#)
 4. (Other topics?)
- (30 mins) Planning for next week
 - Review the [Backlog Board](#) - Are there any tickets here stuck in the "Triage" column? We'd like to keep this column as small as possible.
 - Review the [7.6 Project Board](#) - Assign tickets to developers & assign PRs to reviewers.
 - Paid (by DSpace project) developers must keep in mind priority. If new "high" or "medium" priority tickets come in, developers should move effort off of "low" priority tasks.

- Volunteer developers are allowed to work on tickets regardless of priority, but ideally will review code in priority order

Attendees

- [Tim Donohue](#)
- [Art Lowel \(Atmire\)](#)
- [Andrea Bollini \(4Science\)](#)
- [Paulo Graça](#)
- [Mark H. Wood](#)
- [Grazia Quercia \(4Science\)](#)
- [Corrado Lombardi \(4Science\)](#)
- [Julian Timal \(eScire\)](#)
- [Martin Walk](#)
- [Melissa Anez](#)

Current Work

Project Board

DSpace 7.6 Project Board: <https://github.com/orgs/DSpace/projects/23>

To quickly find PRs assigned to you for review, visit <https://github.com/pulls/review-requested> (This is also available in the GitHub header under "Pull Requests Review Requests")

New Feature development process for 7.6

Per a decision of DSpace Steering, **new features for 7.6 are only welcome if they used to exist in 6.x**. Everything else must be a *fix* that would normally be acceptable in a "bug fix only" release.

7.6 is a "transition" release, where we are transitioning back to our [release numbering scheme](#). As of 8.0, new features will only be allowed in major releases (8.0, 9.0, 10.0) and minor releases will only include bug/security fixes (8.1, 8.2, 8.3).

- **For brand new UI features**, at a minimum, the UI ticket should contain a description of how the feature will be implemented
 - If the UI feature involves entirely new User Interface interactions or components, *we recommend mockups or links to examples elsewhere on the web*. (If it's useful, you can create a Wiki page and use the [Balsamiq wireframes](#) plugin in our wiki)
 - Feature design should be made publicly known (i.e. in a meeting) to other Developers. **Comments/suggestions must be accepted for TWO WEEKS, or until consensus is achieved (whichever comes first)**. After that, silence is assumed to be consent to move forward with development as designed. (The team may decide to extend this two week deadline on a case by case basis, but only before the two week period has passed. After two weeks, the design will move forward as-is.)
 - This does mean that if a UI feature is later found to have design/usability flaws, those flaws will need to be noted in a bug ticket (to ensure we don't repeat them in other features) and fixed in follow-up work.
- **For brand new REST features (i.e. new endpoints or major changes to endpoints)**, at a minimum we need a REST Contract prior to development.
 - REST Contract should be made publicly known (i.e. in a meeting) to other Developers. **Comments/suggestions must be accepted for TWO WEEKS, or until consensus is achieved (whichever comes first)**. After that, silence is assumed to be consent to move forward with development. (The team may decide to extend this two week deadline on a case by case basis, but only before the two week period has passed. After two weeks, the design will move forward as-is.)
 - This does mean that some REST features may need future improvement if the initial design is found to later have RESTful design flaws. Such flaws will need to be noted in a bug ticket (to ensure we don't repeat them in other features) and fixed in follow-up work.
 - **REST API Backwards Compatibility support**
 - During 7.x development, we REQUIRE backwards compatibility in the REST API layer between any sequential 7.x releases. This means that the 7.1 REST API must be backwards compatible with 7.0, and 7.2 must be compatible with 7.1, etc.
 - However, deprecation of endpoints is allowed, and multi-step 7.x releases may involve breaking changes (but those breaking changes must be deprecated first & documented in Release Notes). This means that it's allowable for the 7.2 release to have changes which are incompatible with the 7.0 release, provided they were first deprecated in 7.1. Similarly, 7.3 might have breaking changes from either 7.1 or 7.0, provided they were deprecated first.
 - After 7.x development, no breaking changes are allowed in minor releases. They can only appear in major releases (e.g. 7.x8.0 or 8.x9.0 may include breaking changes).
- **No new Entity Types will be accepted in 7.x**
 - Because new out-of-the-box Entity Types require strategic planning, we have decided that we will be unable to accept new Entity Types in any 7.x release. That said, any newly suggested Entity Types will be passed along to Steering / Leadership so that they may be considered during the planning of the 8.0 release.
 - Enhancements, improvements or bug fixes to the Configurable Entities feature itself, or existing out-of-the-box Entity Types are still welcome in 7.x. We want to ensure that Configurable Entities is made as stable and usable as possible in 7.x, in preparation for discussions of new entity types in 8.x and beyond.

Issue Triage process for 7.6

- **Overview of our Triage process:**
 1. **Initial Analysis:** [Tim Donohue](#) will do a quick analysis of all issue tickets coming into our [Backlog Board](#) (this is where newly reported issues will automatically appear).
 2. **Prioritization/Assignment:** If the ticket should be considered for this release, [Tim Donohue](#) will categorize/label it (high/medium/low priority) and immediately assign to a developer to further analysis. Assignment will be based on who worked on that feature in the past.

- a. "high priority" label = A feature is badly broken or missing/not working. These tickets must be implemented first, as ideally they *should be resolved* in the next release. (Keep in mind however that priorities may change as the release date approaches. So, it is possible that a "high priority" ticket may be rescheduled if it is a new feature that cannot fit into release timelines.)
 - b. "medium priority" label = A feature is difficult to use, but mostly works.. These tickets *might* be resolved prior to the next release (but the release will not be delayed to fix these issues).
 - c. "low priority" label = A feature has usability issues or other smaller inconveniences or a non-required feature is not working as expected. These tickets are simply "nice to have" in the next release. We'll attempt to fix them as time allows, but no guarantees are made.
3. *Detailed Analysis*: Developers should immediately analyze assigned tickets and respond back within 1-2 days. The developer is expected to respond to [Tim Donohue](#) with the following:
- a. Is the bug reproducible? (If the developer did not understand the bug report they may respond saying they need more information to proceed.)
 - b. Does the developer agree with the initial prioritization (high/medium/low), or do they recommend another priority?
 - c. Does the bug appear to be on the frontend/UI or backend/REST API?
 - d. Does the developer have an idea of how difficult it would be to fix? Either a rough estimate, or feel free to create an immediate PR (if the bug is tiny & you have time to do so).
 - e. Are you (or your team) interested in being assigned this work?
4. *Final Analysis*: [Tim Donohue](#) will look at the feedback from the developer, fix ticket labels & move it to the appropriate work Board. If it is moved to the [Project Board](#), then the ticket may be immediately assigned back to the developer (if they expressed an interest) to begin working on it.
- a. If the ticket needs more info, [Tim Donohue](#) will send it back to the reporter and/or attempt to reproduce the bug himself. Once more info is provided, it may be sent back to the developer for a new "Detailed Analysis".

Notes