

Project: Auto-Populating Release Documentation

- [Project Overview](#)
 - [Dynamic Transclusion Architecture](#)
 - [Pre-Development Challenges](#)
 - [Solution](#)
 - [Development Overview](#)
 - [Development Details](#)
 - [Achieved Outcome](#)
 - [Post-Development Challenges](#)
- [Jira](#)

Project Overview

Design and develop a third-party template and macro for the marketing to the Atlassian software community.

Dynamic Transclusion Architecture

In my most recent role as a Senior (Lead) Technical Writer, I was tasked with defining and solving some persistent issues with the software release documentation processes.

Working in a fast-paced Agile software development environment, the release documentation (release notes, patches, hot fixes and procedures) was produced and/or updated several times every week, sometimes more than once a day.

However, the current process was ad hoc and stale; it did not account for any future scalability, and, too often, the content (a table of related bug entries) tended to be out of date or, at times, even inaccurate.

Pre-Development Challenges

I determined these issues were due mostly to the content gathering: the current process was really no process; it was an amalgam of emails, texts—even sticky notes and word of mouth—that was being used to compile the bug lists and declare them ready for the release documentation. This was compounded by the fact that a number of the Program Managers, Developers, and Tech Writers were based in Bulgaria which added challenges of time and distance, and language and culture.

Solution

What was required to solve the issue was a process that was more automated and fool proof—a “single source of truth” (SSOT) transclusion architecture where the content is stored and managed as a single, unique aggregation, which can only be referenced in any other instance.

Development Overview

Atlassian Confluence (essentially a WYSIWYG shell over HTML/CSS) was used as the documentation production platform specifically for its robust collaboration features (allowing multiple stakeholders to work simultaneously on the same document in real time) and its powerful History features (which can be used to easily revert any mistakes—major or minor—by one stakeholder while retaining any changes by other stakeholders). It also served well as a “headless” content management system (CMS).

Jira was used as the bug base for New Features, Enhancements, Fixed Issues, Known Issues and Limitations.

Confluence and Jira both use Jira Query Language (JQL), so the two platforms “handshake” well.

Because of this, I decided to leverage this feature set to develop a process which removed the human element as much as possible by dynamically populating the Confluence documentation templates with the latest SSOT (table of related bugs).

A high-level design (HLD) package, including a flowchart of the proposed process and a persona-based workflow story, was created and distributed to stakeholders for review. After a series of revisions based upon the feedback, a full-feature paper user experience (UX) wireframe analog was produced (this included over 200 possible interaction variants).

Development Details

A set of required "Developer Notes" fields were added in the Jira tickets; if a required Developer Notes field had not been given a value, Jira would return an error informing the user that the field should be filled (the developer cannot move forward without entering a value). The Developer Notes fields included:

- Steps to Reproduce the Issue
 - Detailed Results
 - Expected Results
 - Actual Results
- Root Cause
- Fix
- Affected Areas
- Dependencies
- Workaround.

Additionally, a checkbox trigger (flag) was added named "Include in Release Documentation?" Selecting this exposed a drop-down menu offering the following options (required):

- New Features
- Enhancements
- Fixed issues
- Known Issues
- Limitations.


After choosing a release documentation option, a set of "Release Documentation" fields is exposed; these are clones of the related Developer Notes fields. These cloned Release Documentation fields are available for the tech writer to edit as needed without corrupting the original Developer Notes information.

The Release Documentation fields were the SSOT and could ONLY be changed or updated through the Jira ticket, by the assigned stakeholder(s).

 If a Developer's Note entry is updated, the related Release Documentation field is automatically updated with that change as well.

The next step was the development of an XML-based macro that pulls (transcludes) the SSOT (Jira item) into an auto-generated table in the Confluence documentation template.

Using JQL formulas (Jira item key + release version, etc.) determined which bugs went to which documents as well as where in those documents (various options were included to add custom columns for customer links, Epic links and so forth).

 At this point, if an SSOT entry is updated in Jira, refreshing the Confluence template page would automatically update the template with that change as well.

Achieved Outcome

After a few rounds of testing and few tweaks (mostly tool tips and error messaging), the system was performing as expected; a fully automated, dynamically driven process for populating the release documents with up-to-date single-sourced content was in place:

Jira [Developer -> Tech Writer -> SSOT] -> [macro] -> Confluence [document template -> CMS] -> PDF/Web page -> End User

This process ultimately saved countless man hours and greatly improved technical accuracy.

New Features Roku Client 2.5.0 FT2	
New Features ID	Details
ROK-1480	From the User Profile page, user can activate video event and error monitoring. Press UP/OPTION/UP/OPTION/UP to enter monitoring mode. Set user options to control the display. Real time statistics will show during playback, and error code details will show on video errors.
Enhancements Roku Client 2.5.0 FT2	
Enhancements ID	Details
ROK-979	When a playback resourced has the HAS_ADVERTISING tag returned by PRM then the client should add extra parameters to the playback url per specification.
ROK-1478	Improved support for speech-to-text input for most screens. Not supported on User Name screen when creating new profile.
ROK-1479	PRM is used only when PRM service is directly assigned to the customer account.
Fixed Defects Roku Client 2.5.0 FT2	
Fixed Defects ID	Details
ROK-1468	Release Notes Description Updated EPG navigation to ensure currently playing program is always selected when entering EPG. When navigating with EPG and selecting programs and viewing Detailed Info Page, returning to EPG maintains focus on the last selected program. Fix Updated logic to correctly preserve navigation history Affected Areas EPG



Example Auto-Generated SSOT Output (from an actual release document)

Post-Development Challenges

A workaround was needed to account for any Known Issues or Limitations in one release becoming Fixed Issues in a future release (Known Issues and Limitations needed to be persistent as such in the release history). I used a set of labels ("Include_in_Known_Issues" + [the persistent release version]) to account for this, which was not the most elegant way to do it, but it did allow any stakeholder to easily update a bug to "fixed".

Bonus challenge: People are naturally averse to change. Event though this system greatly improved both ease of production and accuracy of content, it took a number of releases to convince people of its efficacy (particularly in Eastern Europe). Once it finally took hold, it became a Standard Operating Procedure (SOP) throughout the company.

Jira

Type	Key	Summary	Include in Release Documentation?
	APRD-2	Design and develop a third-party macro-based system for auto-popula...	
	APRD-1	Develop concept and HLD for APRD	Release Notes
2 items			Synced just now 