John Paz

Señor Content Design

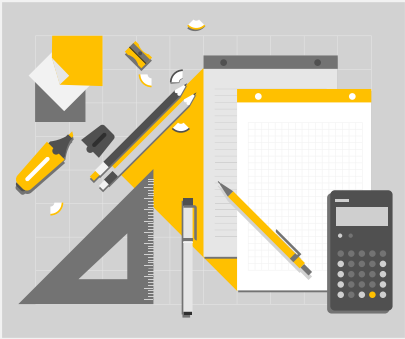
Content Design Portfolio

Updated March 2021

About Me

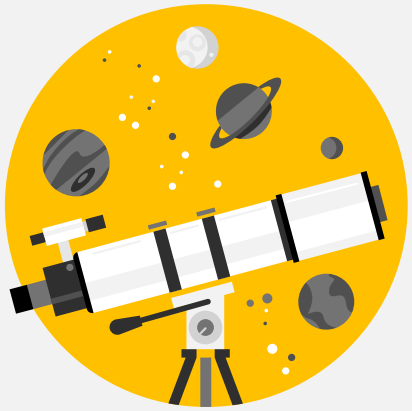
Specialize in technical content

I have a passion for simplifying complex information experiences.



15 years of experience

UI copy reviews, user research, and strategic design thinking.



Great in Agile teams on the product triad

I love getting involved early in the dev lifecycle (I'm also a Scrum Master).

Powerful storyteller and public speaker

I rely on my diverse work/life experiences and storytelling ability to advocate for users.

Work Examples

UX Writing

Menlo Connect Installer

2021

Menlo Connect Installer

FORMAT/TOOLS

- Google Docs
- Figma
- Snag-It
- Pendo

UX WRITING

USER RESEARCH

DESIGN THINKING

DOCUMENTATION

About the project

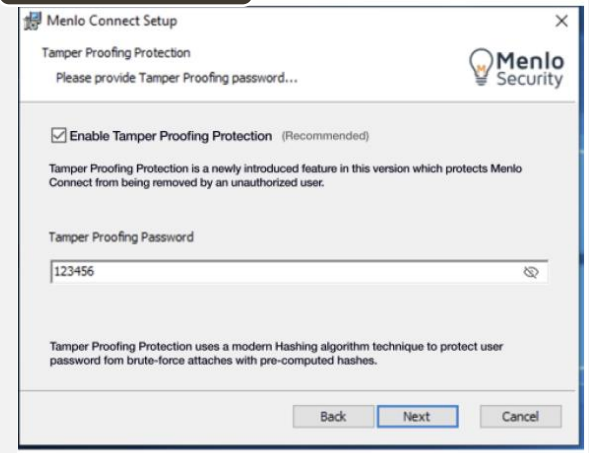
Adding the Tamper Proofing/Uninstall Protection feature to the Menlo Connect installer.

My contribution: I reviewed this screen (and the others in the flow) for clarity, and I insisted on changing the name of a new enhancement for better accuracy and clarity.

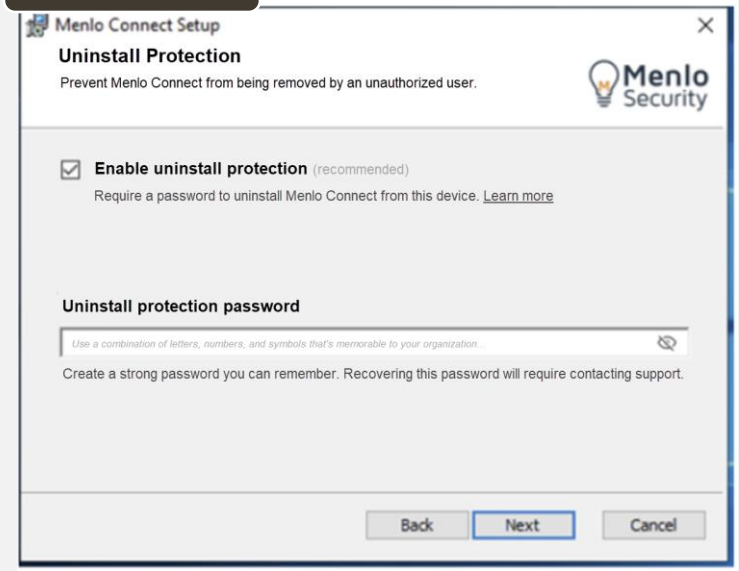
My suggestions for improvement:

1. **Change from "Tamper Proofing Protection" → "Uninstall Protection"**
    1. The name felt misleading; the device could still be tampered with in other ways.
    2. "Uninstall Protection" is more descriptive and clearer.
2. **Rewrite field labels and descriptions**
    1. Keep the goal of this screen in mind – helping admins set an uninstall password.
    2. Remove the word "please," it detracts from the authority of the copy.
    3. Link to documentation, where there will be more details about password recovery.
    4. Add placeholder text to encourage information security best practices.
    5. Warn users about the consequences of losing the password (but don't scare them).
3. **Collaborate with Support regarding password recovery procedures.**
    1. Make support aware of the changes to the screens and addition of feature.
    2. Help document the internal password recovery knowledge base article.

Original



Improved



2021

Menlo Connect Installer

FORMAT/TOOLS

- Google Docs
- Figma
- Snag-It
- Pendo

UX WRITING

USER RESEARCH

DESIGN THINKING

DOCUMENTATION

Process

Scope

1. **Go where the work is.**
   The dev teams don't always know what kind of support writers provide, so it's important to listen.
2. **Know the schedule.**
   I booked a meeting with the Director of Product Management to better understand the event and audience.
3. **Get access.**
   Access to the tools isn't always possible, so I made sure I knew which design documents and mockups to use.

Iterate

1. **Templates first.**
   The teams insist on using Google Docs, but none were used for copy review before. I made templates.
2. **Heavy on detail.**
   I had to go into more details than was typical to make sure the names of components was clear. Dev team were in another time zone.
3. **Check in early and often.**
   With a wobbly source of truth, I had to check in early and often to make sure I had the latest and my suggestions were valid.

Review

1. **KIT with POCs.**
   With so many actors with crisscrossing time zones involved, knowing who to ask, and when, was vital.
2. **Test when you're able.**
   Later in the release cycle, an RC build will become available, and I would use it to validate my suggestions were correct; proved invaluable.
3. **Meetings when in doubt.**
   As time became scarce it became necessary to schedule face-to-face reviews with devs to speed up reviews.

Revise

1. **Know what's important.**
   Because I was involved late in the process note everything got done; some went to backlogs.
2. **Get support involved.**
   Get Support involved. They can provide reviews and add vital context for tricky situations. They also help identify lagging problem indicators.
3. **Keep the doc updated.**
   The UI changes meant some of the doc was now outdated. I ensured the work was addressed or put in a backlog.

Team

- 1 Writer (Me)
- 2 Product Managers
- 4 Engineers (contractors)
- 1 Support Engineer

Project Duration

21 days

UX Writing/User Research

Bitbucket Cloud CoreX Copy Audit

2019

Bitbucket Cloud CoreX Content Audit

FORMAT/TOOLS

- Mural
- Confluence pages
- Jira issues
- Bitbucket Pull requests

UX WRITING

USER RESEARCH

DESIGN THINKING

DOCUMENTATION

Project Overview

- **Problem statement**: There were reports that things were harder to find, it was difficult to complete tasks, and it was difficult to predict where to find help. Evidence of this was repeat support cases from new users, negative sentiment reports on documentation feedback forms, and new customer churn.
- **My contribution**: I identified over 200 usability bugs and improvements, scoped the amount of effort and value for each, then aggregated and scoped the work for a small team of part-time contract engineers to work on for two quarters.

Project Goals

Reduce new user churn

Create style guides and patterns. Establish consistency among content elements and components.

Harden the core user experience. UI inconsistencies contributed to customer dissatisfaction and churn.

Create a visual journey map. Identify the highest priority screens and dialogs every user encounters.

Reduce complexity perceptions

Triage feedback and support cases. Things were hard to find, start-up tasks opaque, and hard to find help.

Keep accessibility in mind. Included concerns for screen readers and visually impaired users.

Scope, chunk, and plan dev work. Identified quick wins, design the solution, review the changes.

2019

Bitbucket Cloud CoreX Content Audit

FORMAT/TOOLS

- Mural
- Confluence pages
- Jira issues
- Bitbucket Pull requests

UX WRITING

USER RESEARCH

DESIGN THINKING

DOCUMENTATION

Research Process

Scope

1. **Define the problem.**
   Audits can encompass anything, so it was important to understand intended outcomes.
2. **Find the time.**
   This project was done in addition to my other responsibilities. I had to carve out time to work on this.
3. **Gather resources.**
   We probed backlogs of user feedback and feature requests to inform our work (and not reinvent the wheel).

Iterate

1. **Weekly meetings.**
   The designer and I met weekly during the journey map creation process, which was very helpful.
2. **Cataloging severity.**
   I never had continuous time for this project, so leaving detailed notes helped keep important tasks top of mind.
3. **Chunking.**
   Once themes began to emerge, grouping similar types of changes helped with velocity of the impact of changes.

Review

1. **Present findings.**
   Stakeholder buy-in to get approval for developer time, so making findings clear was important.
2. **Internal blog post.**
   Since this was a long-running project, a narrative-style blog post helped convey the work involved and how to contribute.
3. **Coordinate with product.**
   Lots of the changes we proposed coordinated with planned feature development for greater efficiency.

Measure

1. **Coordinate with dev.**
   All the hard work would amount to little without contract engineers making the changes.
2. **Put metrics in place.**
   We had to always be asking the devs, “How do we know that worked?” to put in place ways to measure user interaction.
3. **Team reports.**
   My Jira work paid off in the form of reports on how much work went into this project, recognizing everyone’s effort.

Team

- 2 Writers (Me + another)
- 1 Designer
- 3 Engineers (contractors)
- 1 Product Manager

Project Duration

9 months

2019

FORMAT/TOOLS

- Mural
- Confluence pages
- Jira issues
- Bitbucket Pull requests

UX WRITING

USER RESEARCH

DESIGN THINKING

DOCUMENTATION

Bitbucket Cloud CoreX Content Audit

Research Process

Scope

1. **Define the problem**: audits can encompass anything, so it was important to understand intended outcomes.
2. **Find the time**: this project was done in addition to my other responsibilities. Carving out time to work on this was crucial.
3. **Gather resources**: we probed backlogs of user feedback and feature requests to inform our work (and not reinvent the wheel).

Iterate

1. **Weekly meetings:** the designer and I met weekly during the journey map creation process, which was very helpful.
2. **Cataloging severity:** I never had continuous time for this project, so leaving detailed notes helped keep important tasks top of mind.
3. **Chunking:** Once themes began to emerge, grouping similar types of changes helped with velocity of the impact of changes.

Review

1. **Present findings:** stakeholder buy-in to get approval for developer time, so making findings clear was important.
2. **Internal blog post**: Since this was a long-running project, a narrative-style blog post helped convey the work involved and how to contribute back.
3. **Coordinate with product:** lots of the changes we proposed coordinated with planned feature development for greater efficiency.

Measure

1. **Coordinate with dev:** All the hard work would amount to little without contract engineers making the changes.
2. **Put metrics in place**: We had to always be asking the devs, “How do we know that worked?” to put in place ways to measure user interaction.
3. **Team reports**: My diligent Jira work paid off in the form of reports on how much work went into this project, recognizing everyone’s effort.

Team

- 2 Writers (Me + another)
- 1 Designer
- 3 Engineers (contractors)
- 1 Product Manager

Project Duration

9 months

2019

Bitbucket Cloud CoreX Content Audit

FORMAT/TOOLS

- Mural
- Confluence pages
- Jira issues
- Bitbucket Pull requests

UX WRITING

USER RESEARCH

DESIGN THINKING

DOCUMENTATION

Outcomes

About the journey map:

- **Collaborate with design:** a designer and I met one a week to discuss and iterate on the document.
- **CoreX journey:** This journey map that outlines the core experiences all Bitbucket Cloud users go through.
- **We used Mural to create the document**: collaborating asynchronously online, it became a living document we could depend on.

2019

Bitbucket Cloud CoreX Content Audit

FORMAT/TOOLS

- Mural
- Confluence pages
- Jira issues
- Bitbucket Pull requests

UX WRITING

USER RESEARCH

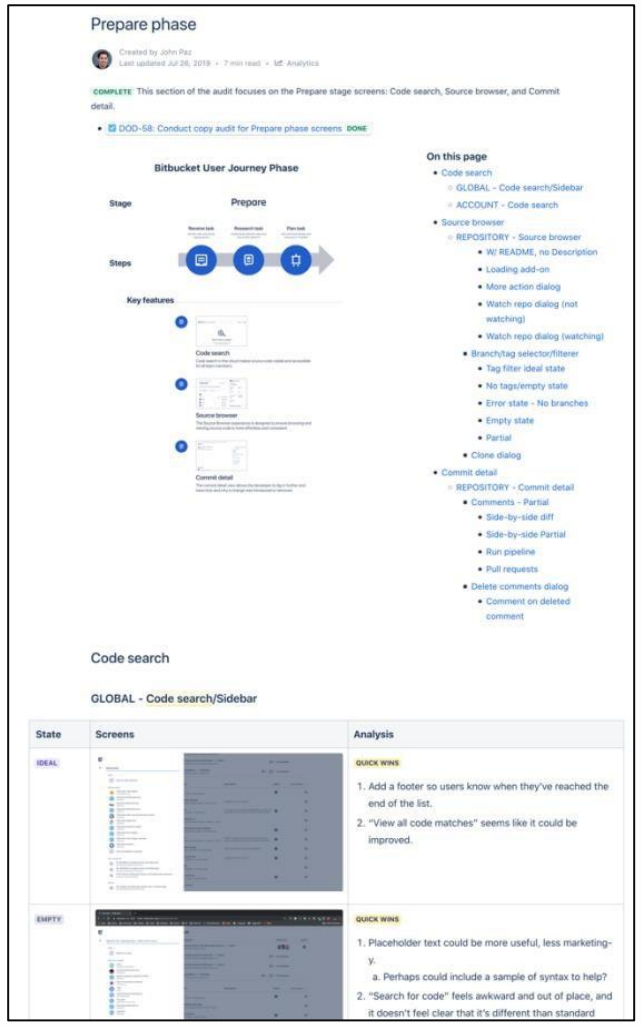
DESIGN THINKING

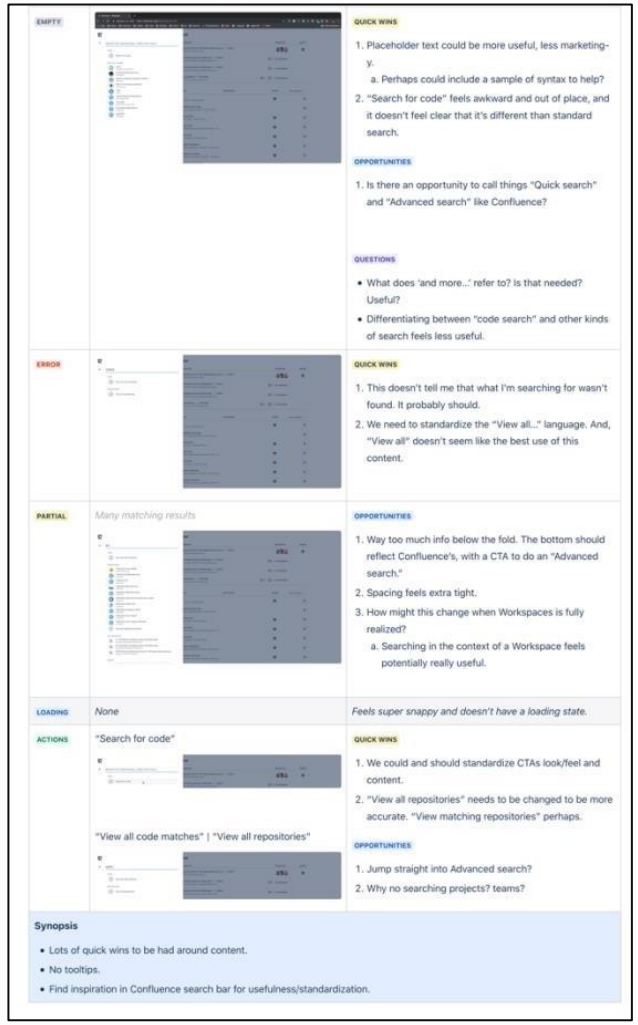
DOCUMENTATION

Deliverables

About the content audit:

- **I used Confluence pages to conduct the audit**: I systematically worked my way through the interface taking screenshots and evaluating areas for improvement.
- **I created Jira issues to track the dev work:** making mockups and providing the changes to make as requirements within the issues.
- **Not pictured**:
  - **200+ Jira issues**: We identified hundreds of bugs, usability and accessibility problems.
  - **Internal blog post**: an internal team blog post to communicate findings and get feedback.
  - **Mockups and wireframes**: to communicate changes, and attached to Jira issues for dev.
  - **Pull requests**: reviewing dev changes via PRs weekly.





Documentation

Bitbucket Cloud API Proxy

2018

FORMAT/TOOLS

- Jekyll static site generator
- Postman
- Confluence pages
- Visual Studio
- Markdown
- Bitbucket pull requests

UX WRITING

USER RESEARCH

DESIGN THINKING

DOCUMENTATION

Bitbucket Cloud API Proxy

Project Overview

- The goal of the project was to describe to add-on developers how to use the new API proxy module to optimize their integrations and adhere to new GDPR requirements.
- **Constraints**: The change was reactionary to GDPR, and there were some architecture requirements that were originally unknown, and the project took much longer than expected.

Project Goals

Document the API module

Primary goal

Add new sections: The developer doc was out of date. Needed to update the page tree to add content.

Uncover missed dependencies: This started as an internal tool, and it was not ready for production yet. I needed to determine how a developer new to Bitbucket might use this.

Learn to contribute to dev docs: This was my first role working on dev docs. A big part of the project was familiarizing myself with the authoring environment.

Address documentation gaps

Secondary goal

Catalog and track gaps: The API module changes cascaded to other parts of using the API, which required updating other docs.

Schedule dev work: Some of the gaps I found required changes to the codebase, blocking the doc changes. This required coordination with product management.

Draft the missing docs: Despite being out of scope for the original project, I drafted outlines and rough drafts of the missing docs to be reviewed and implemented later.

2018

Bitbucket Cloud API Proxy

FORMAT/TOOLS

- Jekyll static site generator
- Postman
- Confluence pages
- Visual Studio
- Markdown
- Bitbucket pull requests

UX WRITING

USER RESEARCH

DESIGN THINKING

DOCUMENTATION

Deliverables

About the API proxy module document:

- **Zero prior experience on dev docs**: the API proxy module was my introduction to developer docs.
- **Great-looking graphics**: I insisted on the graphic to help portray the information flow. The architecture drew out a crude line diagram that I iterated into a polished into a production ready publishable graphic.

API proxy module (developer.atlassian.com/cloud/bitbucket/proxy-module/)

Bitbucket Cloud Developer | Guides | Reference | Resources

Latest updates

INTRODUCTION AND BASICS
- Integrating with Bitbucket Cloud
- Getting started
- Frameworks and tools
- Install an app from your site

SECURITY
- Security overview
- Understanding JWT for apps
- Authentication for apps
- OAuth 2.0
- Query string hash

LEARNING
- Patterns and examples
- API proxy module
- Object hydration
- Application properties

Last updated Jan 7, 2021 — Rate this page:

API proxy module

The API proxy module creates a layer between your API and Bitbucket to provide the heavy lifting for authentication, parameter substitution, object hydration, and other services. This should make the API integrations and apps you build into Bitbucket Cloud more efficient and easier to build.

You create the proxy connection when you map a URL pattern to a specific destination URL in your app descriptor. The simplest form of this is:

```
"modules": {
    "proxy": {
        "/proxy-example/{target_user}/{repository}": {
            "destination": "/proxy-example?name={repository.owner.display_name}", // required
            "methods": { ... }, // optional
            "conditions": { ... }, // optional
            "scopes": { ... } // optional
        }
    }
}
```

Here `/proxy-example/{target_user}/{repository}` is the path available on Bitbucket, and would be the path you would use in calls to an `AP` object, for example. The destination URL is a path served by your server, and maps to a specific Bitbucket resource, in this case the display name of the repository owner.

Before you begin

It's important to understand these core concepts before you jump in to using the API proxy module.

- Authentication between Bitbucket and Connect apps
- Authorization, including conditions and scopes
- Context parameters in path segments

Looking for reference doc or examples?

Reference: Find the elements, definitions, and properties of the API proxy module in the API proxy module reference docs.

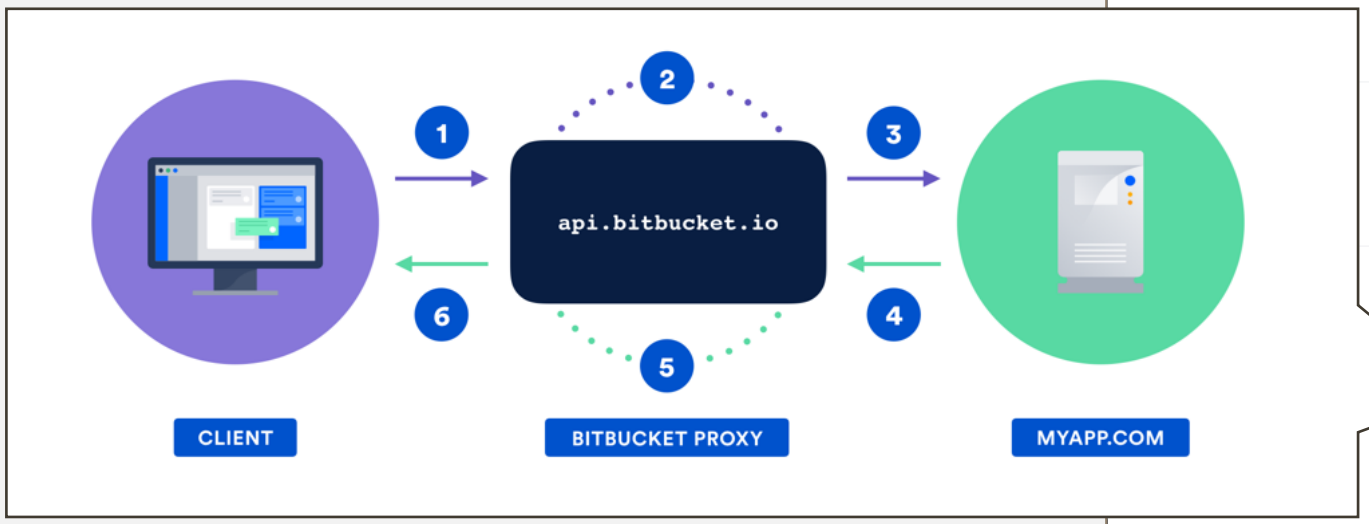
Examples: Example configurations that demonstrate how to use the proxy module in your app descriptor can be found in the Examples section.

How it works

The following diagram depicts the basic operation of proxy module, from authenticating the client call to hydrating the response object.

ON THIS PAGE
- How it works
- Examples
- Object hydration
- Restrict HTTP methods

Data flow of the proxy module



2018

Bitbucket Cloud API Proxy

FORMAT/TOOLS

- Jekyll static site generator
- Postman
- Confluence pages
- Visual Studio
- Markdown
- Bitbucket pull requests

UX WRITING

USER RESEARCH

DESIGN THINKING

DOCUMENTATION

Deliverables

About the object hydration document :

- **Originally part of the proxy module doc**: This entire document was meant to be part of the API proxy module documentation; it's actually longer!
- **Required extensive paring with a dev:** Over several meetings we established requirements for the ideal set of examples to demonstrate all the vital functionality without confusing people.

Object hydration (developer.atlassian.com/cloud/bitbucket/proxy-object-hydration/)

Bitbucket Cloud Developer | Guides | Reference | Resources

Latest updates

INTRODUCTION AND BASICS
- Integrating with Bitbucket Cloud
- Getting started
- Frameworks and tools
- Install an app from your site

SECURITY
- Security overview
- Understanding JWT for apps
- Authentication for apps
- OAuth 2.0
- Query string hash

LEARNING
- Patterns and examples
- API proxy module
- **Object hydration**
- Application properties

PRIVACY GUIDELINES
- User privacy guide for app developers
- Profile visibility

BUILDING BLOCKS
- App context
- App descriptor
- Conditions
- Context parameters
- Access to context parameters

OTHER CONSIDERATIONS
- Atlassian Design Guidelines
- Atlaskit
- Atlassian Marketplace
- Cloud app licensing

Last updated Jan 7, 2021 — Rate this page:

Response object hydration

Object hydration allows your app to respond to proxied API requests with a minimal amount of data for a resource and Bitbucket will add the full object from the Bitbucket database. This API proxy service will automatically fill in any missing elements of certain Bitbucket data types returned by a remote service. This allows remote services to keep track of only the absolute minimum information to uniquely identify an object.

For example, if your app were meant to get details of a single repository it only has to store a way to identify the repository, which can be as little as a repository UUID. Without using the proxy module, a simple request using a repository UUID returns a partial repository response object, without being hydrated, that would look like this:

```
1 {
2   "type": "repository",
3   "uuid": "{aa5acc33-e5f7-43e9-883d-50325fc68ca8}"
4 }
```

Using the proxy module, when the response is passed to Bitbucket's API proxy, the proxy:

1. Decodes the JSON from the remote service.
2. Scans the object graph for type elements defined in Bitbucket.
3. For each type, queries the database and substitutes it for the full object.
4. Serializes the resulting object, producing layouts consistent with other, core APIs.

The hydrated response coming from the API proxy provided to the client would look like the entire response body from the repository endpoint.

```
1  {
2      "scm": "git",
3      "website": "",
4      "has_wiki": false,
5      "uuid": "{aa5acc33-e5f7-43e9-883d-50325fc68ca8}",
6      "links": {
7          "watchers": {
8              "href": "https://api.bitbucket.org/2.0/repositories/atlassian_tech_writing/teams-in-spa
9          },
10         "branches": {
11             "href": "https://api.bitbucket.org/2.0/repositories/atlassian_tech_writing/teams-in-spa
12         },
13         "tags": {
14             "href": "https://api.bitbucket.org/2.0/repositories/atlassian_tech_writing/teams-in-spa
15         },
16         "commits": {
17             "href": "https://api.bitbucket.org/2.0/repositories/atlassian_tech_writing/teams-in-spa
18         },
19         "clone": [
20             {
21                 "href": "https://bitbucket.org/atlassian_tech_writing/teams-in-space-tutorial-conte
22                 "name": "https"
23             },
24             {
25                 "href": "git@bitbucket.org:atlassian_tech_writing/teams-in-space-tutorial-content.
```

ON THIS PAGE
- Capturing the parent object of the object you're capturing
- Response objects that can be hydrated
- Issue schema
- Snippet schema
- Pull request schema
- Repository schemas
- Account schemas
- Project schemas
- Issue comment schema
- Commit comment schema
- Pull request comment schema
- Build schemas
- Commit schema
- Snippet commit schema
- Commit file schema
- Commit directory schema
- Rendered schemas
- Error schema

Capturing the parent object of the object you're capturing

For objects that belong to another object -- like how a repository belongs to a user or team -- **the parent object may need to be resolved in order to resolve a child object**

2018

Bitbucket Cloud API Proxy

FORMAT/TOOLS

- Jekyll static site generator
- Postman
- Confluence pages
- Visual Studio
- Markdown
- Bitbucket pull requests

UX WRITING

USER RESEARCH

DESIGN THINKING

DOCUMENTATION

Documentation Process

Scope

1. **Listen, ask questions**: The original project was daunting, so I had to listen and do a lot of background reading.
2. **Create an outline**: This project had a lot of interdependent and interweaving parts, an outline helped stay organized.
3. **Shared understanding**: Resist the urge to 'figure it out later' when the initial assignment only consists of 'document this' for requirements.

Iterate

1. **Get out of the hole:** Instead of iterating alone, it was important to get reviewers eyes on the work as soon as possible.
2. **Raise concerns early:** As soon as the scope starts to creep, raise the red flag and ask for support or push back the deadlines.
3. **Feedback from outside the team:** Sometimes your teammates are too close to the problem and share biases. Look for reviews from other teams.

Review

1. **Schedule stakeholder reviews: T**he folks reviewing my work get busy, so I make sure to schedule their reviews..
2. **Be specific**: Long pieces of documentation like this can be daunting to review. Be precise about what should and shouldn't be reviewed.
3. **Test it yourself:** Though it can add to turnaround time, testing procedures yourself is the only way to ship instructions with genuine confidence.

Measure

1. **Use your resources:** Not everyone reads release notes, so it can help to advertise new docs in community forums.
2. **Set a future date**: We don't always stop to smell the roses. We also don't plan to stop and measure work in the future but doing so ensures you will.
3. **Check social media**: If the product is publicly available and popular chances are someone online has shared an opinion of it. Find them.

Team

- 1 Writers (Me)
- 1 Product Manager
- 3 Engineers
- 1 Software Architect

Project Duration

4 months

More About Me

A few quick things to note

More About Me

Public Speaking

paz.tips/john-presentation-building-belonging

More About Me

Diversity

I'd Rather Be Writing
exploring technical writing trends and innovations

Home | API documentation course | Podcasts | About | Archives | Series | Misc. | Nav | Search all content

Stay updated

Keep current with the latest trends in technical communication by subscribing to the I'd Rather Be Writing newsletter. *5,600+ subscribers. (See email archive here.)*

email address

Subscribe to newsletter

Diversity in tech comm -- Conversation with John Paz

by Tom Johnson on Jun 12, 2020 · 3 comments
categories: technical-writing

Summary: In this conversational Q&A post, I chat with John Paz, a senior content designer for Atlassian, about diversity in tech comm. It's well known that the tech industry, particularly in Silicon Valley, struggles to live up to its ideals about diversity. The employee demographics at most tech companies don't reflect the same demographics of their surrounding communities. Even when minorities are hired, promotions and leadership positions within the company are another roadblock. John gives us his unique insight into the diversity issue, specifically focusing on tech comm and his experiences both in the Bay area and elsewhere.

Contents

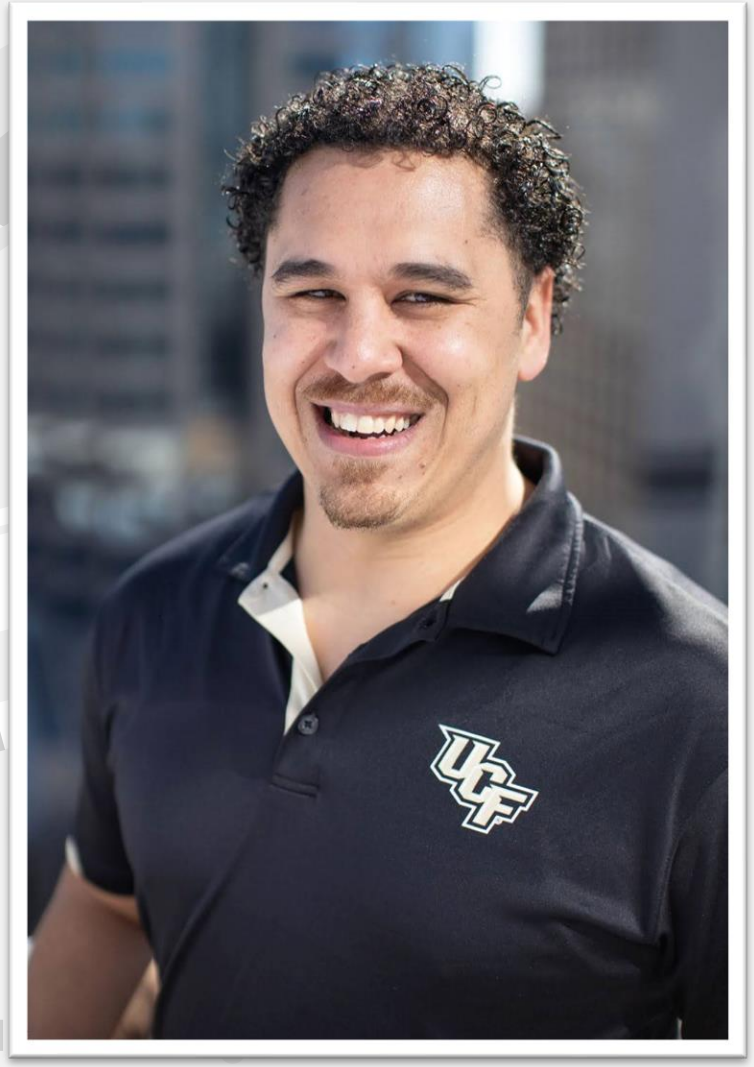
- Discussion articles and summary

paz.tips/john-interview-idratherbewriting2020

More About Me

Diversity

paz.tips/john-interview-pocit2019



Get in touch

Look for more of my work online

paz.tips/johnpaz-linkedin

paz.tips/johns-online-portfolio

paz.tips/twitter-srcontentdesign

Content Design Portfolio

Updated March 2021