A Good Idea at the Time

by
Daniel Messer

Daniel Messer (Tanglewood Hill Studio)
cyberpunklibrarian@protonmail.com
@cyberpunklibrarian@hackers.town

FADE IN:

INT. DAN'S OFFICE - DAY

Dan is sitting at his desk, typing and working on library related things. He finishes up and turns to the camera.

DAN

Back in high school, a friend of mine introduced me to a concept that's stuck with me ever since. "When what you need don't meet your eyes, you must learn to improvise." Now, as it happens, that's a thought technology that comes in handy when you're dealing with libraries and library automation systems.

Oh, yes, introductions and things. Hi! My name is Dan, and I'm a Polaris ILS Administrator for Library Systems & Services, and I'm your friendly neighbourhood Cyberpunk Librarian. I've been working with the Polaris Integrated Library System for over twenty years, and almost half of that time has been spent in some kind of ILS administrative role. Thanks for coming to my virtual talk, or watching my video, or whatever it is that's going on right now. I promise I will not ask you to like and subscribe.

Now, if you've got a notepad or a note app open in front of you, that's fine, but you can sit back and relax with your beverage of choice if you like. I want you to know that all of the things I'm going to talk about in this presentation will be available to you to download and look at later on. That includes a full transcript because this is, after all, a video and there's a screenplay for it. So check out the links at the end of the video and I'll make sure you get everything related to this talk.

CUT TO

DAN'S OFFICE - FULL SHOT

Dan is standing up, leaning against a wall, with a coffee cup in his hand. The coffee mug has a Cyberpunk Librarian logo on it

DAN

Today, we're going to talk about those times when you need Polaris to do a thing, but it doesn't actually do that thing. We're going to solve some of these problems with code and coffee, or whatever beverage you prefer. If you're not a software developer, don't leave! Because hey, I'm not a software developer either. I'm a semi-competent coder who writes semi-competent code. Depending on your situation, you may not need to write a single line of code to do anything. And while I'm going to talk about working through problems with code, I want to primarily focus on attacking those problems from different points of view. Believe me, code isn't the answer to everything. Sometimes it greases the wheels. Sometimes it's a broom handle through the front wheel of your bicycle.

So instead of focusing on code alone, let's turn our attention to problem solving. In the end, my thesis is inspiration. I'd like to encourage you to look at your ILS, think about something you'd like to do with it, and then see if you can make it do that thing.

CUT TO

DAN'S OFFICE

Dan stands in front of a different wall - this one set up for compositing the slide deck.

DAN

Sometimes you need to go beyond Polaris, and that's okay. Polaris has several access points, highways if you like, that allow travel into and out of the system. I won't pretend to know your library's circumstances and budget, but I'm willing to bet you paid good money for Polaris, so should never feel bad about getting more than your money's worth. Sometimes, you need an answer specific to you. It's quite possible that no one but you and your library has this problem and, if you solve it, you’re the hero.

And sometimes, you just want to poke the box. See what happens. Explore what’s possible. Because today’s useless bit of information can easily transform into tomorrow’s game changing idea.

Weirdly, for a dude who makes his living working on an ILS so embedded in the world of Microsoft that I’m surprised the software doesn’t smell like Starbucks and Pike Place Market — I’m not a Windows guy. My expertise lies in Linux, macOS, and open source software. Because of that, I subscribe to the UNIX Philosophy of Software but, more than that, I’m a big fan of the first rule: Do one thing well.

When you’re as good at writing code as I am, in other words you can make things work after multiple attempts and lots of swearing, the idea of writing software to do one thing is my core competency. If you need to solve a problem, step back and examine your tools. Because my life has been a strange mix of PBS and punk rock, I grew up watching a lot of *This Old House*. There’s a guy on that show, Norm Abram. He’s a master carpenter and, ya know, I noticed that he’s got a lot of tools and, for the most part, they all do one thing.

Norm isn’t the kind of guy to go hammering nails with the handle of his screwdriver nor will he try and cut lumber with a drill press. Every tool is a singular thing that’s good at one, maybe two jobs. What matters is that Norm has lots of tools and he knows how to use each one to solve problems, which usually involved major renovations of Victorian and Cape Style houses.

When you’re approaching a Polaris issue, keep in mind the tools you have available to you. One single tool might not be important, but your answers might lie in combinations of those tools. That’s why toolboxes exist.

So, let’s look at a problem.

CUT TO

TITLE CARD - TELEPHONEY

CUT TO

MAP OF THE UNITED STATES

A map of the United States composited on screen, with pin drops indicating locations of LS&S libraries.

DAN (V.O.)
I help support a bunch of libraries all across the country. California, Kansas, Texas, Virginia, Tennessee, Florida, and more. And of all the facets of Polaris that rankle my hide, you know what used to be one of the most annoying?

CUT TO

BACK TO DAN STANDING IN FRONT OF PRESENTATION WALL

DAN
Telephony.

Most of the libraries have a local telephony server and that server lives either in or nearby the library itself. There are different ways to get into different servers. If you’re lucky, you can RDP from production to telephony. But sometimes you need to access telephony directly. Or sometimes you need to get in through TeamViewer. Or perhaps it’s not TeamViewer but something *like* TeamViewer. And sometimes something is broken, or a firewall rule has changed and we can’t get in at all…

It can be a mess.

The worst part is, when something bad happens to telephony, it typically fails silently. Calls just… stop. We used to find out about this problem via the help desk and trouble ticket, when librarians started complaining that people weren't getting their telephone notifications.

Now, normally, you could install some kind of monitoring system that watches for service failures. But for some of these telephony servers, we couldn't count on that monitor getting a message out to us because it's behind three firewalls or something. Then there's that old dinosaur server that stops calling out because it

(Gradually getting anger and eventually yelling)

HARD LOCKS AND NEEDS A PHYSICAL REBOOT BECAUSE IT WAS BUILT DURING THE BUSH ADMINISTRATION AND…

SUDDEN CUT TO

TECHNICAL DIFFICULTIES CARD WITH ELEVATOR MUSIC

CUT TO

BACK TO PRESENTATION WALL

DAN

So we can't monitor the servers themselves. Too many what-ifs and wherefores. So what to do? Well, this is the part of the talk where I get into approaching the problem from a different angle.

But first, let's talk about smoke detectors.

Funny thing about smoke detectors. They're absolutely crap when it comes to informing you about a fire in your house or apartment. That's because they're not looking for fire. A smoke detector is a stupid device that has no idea what a fire is, but it has a very good grasp on the concept of smoke. In other words, your smoke detector isn't searching for the cause, it's looking for an effect.

So I wrote a programme that isn't looking for the cause of a telephony outage, it's looking for the result. Because the one thing we can always look at is the

NotificationLog table in the
PolarisTransactions database.

TelePhoney gets its name from the phoney sense of stability you have while your telephony server has been down for the last five days. Rather than rely on the staff to notice and report failures, we rely on TelePhoney to let us know about those problems before library staff realize there’s a problem at all. The basic operation of the programme is simple:

It connects to a production server, runs a simple SQL query to find out when the last phone call went out, does some basic math, and if it produces an answer that shows a call hasn’t gone out recently; it emails us.

No, seriously, that’s it.

CUT TO

SCREENCAST OF TELEPHONY

DAN (V.O.)
While I went ahead and built a browser based UI for TelePhoney, that’s mostly there so I can easily hit up the system at any time and see if there are any outages or delays I need to worry about. I’ve often sent screenshots from the app to let librarians know that their systems are up and running, or at least they’re up and running *now*.

CUT TO

DAN - SITTING AT HIS DESK

DAN
Like I said, I’m a Linux and open source doofus, so that’s what I fell back on when I built this thing. Norm Abram was always reaching for a hammer, or a saw, or a drill or something. He wasn’t using an angle grinder or a TIG welder because those aren’t the tools he uses to build things.

TelePhoney running on a WAMP stack, but I’ve got it documented and coded to run on Linux too. It’s written in PHP and leverages the Microsoft ODBC Driver for SQL Server and the Microsoft Drivers for PHP and SQL Server. All of that alphabet soup means is that there’s some software that allows PHP to talk to the Polaris databases, ask questions, and then do things with the answers.

And twice a day it automatically checks all the production servers to see if there’s any smoke from a telephony server fire. The results have been great, our tickets for telephony issues have dropped to almost nothing. TelePhoney logs outages to a database, so we can use that data to spot servers with more than their fair share of outages and take action to fix that. And we get a better feel for telephony usage. Some libraries just don’t make a lot of phone calls.

CUT TO

TITLE CARD - ECOS

CUT TO

DAN SITTING OR STANDING AT A LIBRARY FRONT DESK

DAN

Okay, right, let’s move on to a completely different problem, one that’s happened to many of us in some way or another. Here’s the scenario:

CUT TO

DAN WORKING IN A LIBRARY ON A COMPUTER

DAN

You’re in the middle of a work day, helping patrons, getting things done like David Allen. All’s right with the world until…

THUNDER AND LIGHTING OUTSIDE THE WINDOW AND THEN THE LIBRARY GOES DARK

DAN

Welp, you’ve got a power outage. Or maybe a network outage. Maybe both. Who knows? All you know right now is that you’re not connected to the ILS. How long will you be down? Was that an explosion? Did that transformer down the street blow up again?

Dan gets close to camera, like he’s getting friendly

DAN

(Conspiratorially)

Yeah, this used to happen to me all the time. There’s a transformer down the street and around the corner and it explodes every six months or so. Actually, it’s been about six months since the last time that happened so we’re probably due for…

Explosion heard before Dan finishes

CUT TO

TELEVISION SNOW

CUT TO

DAN, BACK IN THE DARKENED LIBRARY

DAN

Oh my god, that’s such a lame joke because when was the last time you saw static on your television? Anyway, you’ve got a problem. Patrons are in the building, they want to check things out, and you should probably let them do that.

But, no computers, no network, no Internet. What do?

Well, proper planning prevents pitiful performance and you should know that I never say that phrase in that manner but this is supposed to be a family friendly presentation. But I’m hoping you might have a laptop laying around and it’s got a fully charged battery. But what’s it matter? You can’t get online.

And that’s where ECOS comes in.

CUT TO

SCREENCAST OF ECOS

DAN (V.O.)
ECOS is the Emergency Check Out System. I originally called it PECOS which meant Polaris Emergency Check Out System. You may occasionally see an icon of a cowgirl somewhere around the app or repo and now you know why. While I was shouting at some code that wasn't working, it occurred to me that there's no reason this needs to be some kind of Polaris exclusive. Almost every ILS I know of checks things out to patrons by scanning a card and then scanning the item. So I dropped the P and here we go.

CUT TO

DAN IN DARK LIBRARY - A LANTERN SITS ON THE DESK

DAN
There's still a P in ECOS but it's on the back side of the app. In this case P stands for Python and today's presentation is brought to you by the letter P and the number 005.133. ECOS is a Python based application working with the tkinter library to build the fabulous graphical user interface you see here. It's intentionally simple not only to mimic the Polaris check out experience, but also because I didn't want to do a lot of faffing about in tkinter.

If you're wondering why bother with a desktop app like this, instead of relying on Polaris offline, or Leap offline, or maybe some kind of browser based solution - well those are good questions.

ECOS assumes nothing in that in assumes you're working with nothing. It's working off the basis that you have no Internet connection. It assumes you've got no network connection, so a shared drive on the network is inaccessible. It assumes your library may not have a spreadsheet programme installed at every station and, believe me, that happens. IT Departments aren’t always keen to drop an Office license on every computer and they’re don’t want to bother with LibreOffice or something similar.

And with so many Polaris libraries switching to Leap as their primary access to Polaris, ECOS assumes that you have no access to the Staff Client. And while Leap offline is a thing, unless you’ve preinstalled it, it’s not going to be there for you. And besides, Leap Offline has a few issues of its own that make it just a tad unstable for reliable offline use at this time. Or, at least, that’s my opinion. However, and this becomes important in just a minute, ECOS assumes you have a web browser, even without an Internet connection, because it assumes you’re using a computer made within the last decade or so.

But at the outset ECOS assumes you have a Windows desktop and that you can run software locally on the computer. That’s pretty much the core functionality of Windows so, as long as ECOS is installed, it’ll be there when you need it.

CUT TO

SCREENCAST OF ECOS

DAN (V.O.)

The workflow will be familiar to anyone who’s checked out items using the Polaris Staff Client. You start by scanning a card and then ECOS “locks” that card in as the active patron. Then you scan your patron’s items. When you’re done, hit Enter on the keyboard and you’re ready for the next patron.

Keep in mind, ECOS has no idea who these people are nor does it know anything about the items you’re checking out. It’s just recording data.

There are three buttons here. Backup Database creates a copy of your current database as a separate file. If you're down for more than a half an hour, it's a good idea to backup your database every so often. Redundancy in computing can be a good thing.

Clear Database does exactly what you expect. You're going to be asked twice if clearing the database is what you want to do. Because once it's gone, it's gone. When things settle down and you're back online, you'll want to clear your database to protect privacy and because you don't want to start from an old database next time the system goes down.

But, before you clear the database you'll want to export your checkouts. When you click this button, ECOS creates an HTML file with your patron and items rendered as Code39 barcodes and it'll open that sheet in your default browser. Once again, assuming nothing, the software also figures you might not have a PDF reader on that workstation. When your browser opens the sheet, print it out, open up Leap or the Staff Client, and start scanning your barcodes into the system.

CUT TO

DAN STANDING IN WELL LIT LIBRARY

DAN

I used to work for a library system in Arizona and, one fine afternoon, the transformers exploded in the basement of the building housing our servers. We were down for four days. While ECOS isn't that old, the method is. I had the staff scan checkouts in a spreadsheet, changed the numbers to barcodes, and when we came back up we got to work scanning the sheets into Polaris.

We were caught up from four days of downtime in about two hours. And since the data was scanned into the sheet and then scanned from the sheet into Polaris, there were no errors and no typos.

While this video was in the scripting stage, I’d been working on some bug fixes and tweaks for ECOS. One of the new features I want to include is the ability to export your checkouts as Polaris offline files. So that may be something coming soon or already implemented. Check the links at the end of the presentation to find out! I’ll be just as curious as you are!

CUT TO

TITLE CARD - SWAGMAN

CUT TO

DAN BACK IN HIS OFFICE

DAN

So I mentioned poking the box and seeing what was possible. William Gibson once wrote that the street finds its own uses for things and that’s how I approach fiddling about with Polaris. I encourage you, go play with Polaris whenever you can. Read some docs, see what it can do, discover something, and see if you can work that to your advantage. I mean, yeah, let’s not break the system and accidentally destroy the databases or anything. But maybe there’s some blue-sky thing you can do that might be fun, monotony breaking, or lead to knowledge you can use later. I want to talk about just such a blue sky thing that, like many ideas, was born out of anger.

Because you see, there are few things more overpriced on the planet than Ray-Bans, the BMW M5, and self check out machines.

CUT TO

DAN STANDING IN FRONT OF A SELF CHECK OUT

DAN

One day I was working on a problem with a self checkout. I won’t name and shame because, really, there isn’t a self checkout machine that doesn’t have its own problems and imperfections. The thing about this problem is that it was stupid. It was a stupid problem with an expensive machine built by software designers and engineers who should’ve known better. And while I was hammering on this problem I said something out loud to the guy who shares my office and also happens to be the derpiest pit bull on the planet. And what I said was:

“Given how much they paid for this machine, you’d think it’d work better.”

And that sparked an idea. What if, just what if, you could get a self checkout machine on the cheap? And I’m not talking about an ExpressCheck station or anything like that. The last time ExpressCheck got any love was probably about the same time What Does the Fox Say was burning up the charts.

So my question was, could I make a self checkout system that was cheap, but workable? These are the thoughts that kept me out of the really good library schools.

First: What do we mean by cheap?

Well I know that a self checkout machine can cost $10,000 and up so let’s do a percentage of that. And let’s go with, oh, I dunno, five precent, which is $500.

Well, a Raspberry Pi kit costs around $150. You can pick up a 22 inch monitor for $120. Throw in a keyboard, mouse, and a cheap barcode scanner for maybe another 75 bucks and you’re up to $345. I suppose one could splurge and get a nice touchscreen monitor if they wanted.

The software will have to be inexpensive and by “inexpensive” I mean free. As it happens, free software is a thing I do. So I opened up Visual Studio Code, got myself a brand new bottle of Fireball, and went to work.

CUT TO

DAN AT HIS DESK, VISUAL STUDIO CODE ONSCREEN

DAN

At its core, a self checkout machine is just a fancy barcode scanner connected to your ILS, much like the one on your circ desk. The difference is, we let our patrons use this barcode scanner.

Most of these self checks use SIP2 to talk to Polaris and, ya know, that's fine. SIP2 has been around forever and it works well. I'm often confused by people who beat up on SIP2 because it's "old," as if there's something wrong with that. It's a protocol, it works, does the job pretty well, so what's the problem? Microsoft SQL Server is over 30 years old and no one ever brings that up.

CUT TO

DAN STANDING AT A SELF CHECKOUT MACHINE

DAN

For the most part, the self check workflow is the same from machine to machine. The patron comes up and scans their card. They enter their PIN. They scan their items. They tell the machine they're done. They bounce. Everything else is faff and statistics.

CUT TO

DAN BACK AT HIS DESK

DAN

Great, so we're going to need something that takes in a patron's card, checks to make sure it's valid, and then moves ahead with scanning items and associating them to the card.

Right. Time to get coding!

CUT TO

HACKERTYPER.NET OVERLAID WITH TECHNO MUSIC

CUT TO

DAN BACK AT HIS DESK

DAN
Okay, seriously though. Let me introduce you to Swagman, the cheap self checkout system. All you need is a Raspberry Pi, a monitor, a keyboard, a barcode scanner, and a network connection. While it's designed to work with Polaris, ultimately it's designed to work with SIP2 based communications which theoretically means it'll work with almost any major ILS.

Swagman is built on your traditional LAMP stack of Linux, Apache, MariaDB, and PHP. We're using the PHP here for our server side functionality except for a cheeky bit of Python I threw in there. So I guess this is actually a LAMPP stack? But we'll get that in a moment.

CUT TO

SCREENCAST OF SWAGMAN PATRON AUTHENTICATION

DAN (V.O.)
Swagman runs in a web browser and while I'm not a big fan of Chrome, I *am* a big fan of its kiosk mode. So this is a browser based app that runs in a full screen browser. It's running off a web server locally installed on the Pi, but there's no reason you couldn't set it up to connect to a remote web server.

I am not a front end designer and boy does it show. But I can repurpose open source templates pretty good and that's what you're looking at here. Almost everything can be customized because, after all, it's just pictures and code.

Behind the scenes, you've got a database with all the goodies in it that Swagman needs to talk to Polaris. It's got the IP for the SIP server, the SIP user and password, and that kinda stuff. Remember the workflow? Scan the card, enter the PIN, and click/tap the button to move on.

CUT TO

DAN AT HIS DESK

DAN

Here's where that Python comes in, because I could't find a good way to run SIP commands through PHP. I'm sure it can be done, but I refer you back to the beginning of this talk where I clearly pointed out that I'm a *semi*-competent coder. All of the SIP functionality happens through Python because that was easier for me. So when the card is scanned and the PIN is entered, PHP sends that over to an authentication script that talks to the SIP server, handles the resolutions, and kicks back whatever it needs to PHP to throw on screen.

CUT TO

SCREENCAST OF SWAGMAN CHECK OUT

DAN (V.O.)

Checking items out goes through another Python script that, you guessed it, talks to the SIP server and checks the items out to the active card. Like any other self checkout system, the screen is updated as items are scanned. Once the patron is done checking out, they click or tap the button, and the Swagman sends them a receipt.

CUT TO

DAN AT HIS DESK

DAN

Now, you may be wondering, why not print a receipt? And I'm going to be honest, the answer is one part solarpunk, one part selfish bastard.

For the solarpunk reason - Thermal receipt paper is horrible. Have you ever smelled that stuff? Actually, don’t. Don’t smell receipt paper. It’s probably a carcinogen. But as a dude who worked in circulation for seventeen years I can tell you, so many people just dump the receipt in the trash on their way out the door. You can’t recycle it because it’s covered in weird chemicals. So save yourself some money, and help the environment, and don’t buy a receipt printer.

The selfish reasons? Receipt printers are great when they work. But when they suddenly stop working, it typically turns into a cat and mouse game mixed with a wild goose chase of checking drivers, checking the build on your OS, checking for updates, trying to find what’s changed. Honestly, I think Swagman *gains* reliability because it *doesn’t* use a receipt printer. It means you’ll never have to go to battle over printer issues.

Would anyone want to use Swagman, like in a real library environment? I dunno. I mean, I suppose you could. It needs some more work, but that’s the fun of it. Fiddling around with SIP2 and Polaris and making something work hasn’t been done before. And thanks to Swagman, I know far more about SIP2 than I did before.

And whether or not you’re solving a problem, trying something new, or just messing around… it’s stuff like this which breaks up the day and maybe you learn something that’s useful later on.

CUT TO

DAN’S OFFICE - SECONDARY SHOT

DAN

So if you can, take a little time and play around, y’all. No one said every day has to be the same, ya know?

And thank you so much for coming to my virtual presentation! Check out this link for all kinds of goodies including the software repos for the projects I talked about, further information about the technologies involved, a transcript of this presentation, and whatever else seems like a good idea to throw in there.

If you have any questions, feel free to hit me up! You can email me or follow me in the fediverse. You’ll also find me on the IUG Discord where I’m pretty much hanging out every workday. I mean, I just keep Discord open. But for now, enjoy the rest of the conference, and I’ll see you at the Q&A session a little later on!

THE END

SCENE INDEX