Salesforce Quirks

Stuff that is working as designed, but the design is meh

- Salesforce 2 Salesforce (is horrible)
- Multipicklists and when to use them
- Process Builder picklists bug
- PersonAccounts
- Case Assignment Rules triggering
- Flow's Send Email Core Action + Text Templates
- Permission Sets and Permission Set Groups: Shared values and silly rules
- Polymorphic Owners and field references
- That one Hardcoded Id
- SFXD Presents - Parsing the new Help site is horrible

Salesforce 2 Salesforce (is horrible)

To summarize real quick:

Salesforce to Salesforce, or SF2SF, SF 2 SF, or whatever you want to call it, is a pile of flaming garbage that should have been nuked ages ago.

It:

- uses a shit ton of API requests because it syncs one record at a time
- can't be automated without a trigger ; this means for each record you're supposed to click "share" manually
- even if you do add a trigger it still sucks because you can only decide "in general" which fields to sync, not per case
- there's no conflict resolution. Data is different in target org ? better have a backup because that shit goan get overwritten
- there's no guaranteed timeline. It can be instant or take up to 24 fucking hours. yeap. It's normally instant but your max lead time is 24. fucking. hours.
- did I mention there's no conflict resolution and it will overwrite stuff without warning ?

oh and if you do bidirectional sync you're pretty much asking for hell on earth because well if someone overwrites anything it'll get propagated to all orgs with no double check
IT. IS. HORRIBLE.

The worst is that it seems to be some kind of queue so older messages might not be delivered but some newer might and you have no idea why.

I think the only use case that it’s barely usable in is for Case Comments and even there I have seen timeline fuckups (older case comment coming in after a newer one which looks weird).

Multipicklists and when to use them

https://cdn.discordapp.com/attachments/246568944213819393/610943206590251008/rant_on_mu

Text from image:

sorry
if I don't get pinged I tend to go off do something else
short version is multipicklists suck
they suck hard enough to use them to clean my flat when my hoover's on the fritz.
longer version is that a multipicklist in SF is literally just concatenation values together as text
so if you try to use this later you'll get values concatenated together with ;, not a list
which means it sucks for reporting
it sucks for apex code
it sucks when you want to do a list view filter
etc.
a junction object in the end is pretty much the same thing: you get multiple stuff on one single record
except it can store more info
you can report on it
you can list and sort it
and, if the user REALLY wants that on an object
you can even roll it up to concatenate as string in a field
thereby giving the user what they wanted.
the only reasons to ever use multipicklist for me are if
- you don't intent to report on that field
- it's literally just there to highlight stuff that's useful to the user but nothing else
- using a junction object would put undue stress on the user.

Process Builder picklists bug

If you use PBs a lot, read this.

So I have an org that has a huge PB. No one cares. Fine. But what's interesting is they have a LOT of decision nodes. And today I realized some of the nodes referenced "Select One" as a picklist value. Which of course would never work. Except Process Builder doesn't let you select "Select One" when you save. And a Process Builder is read only when active, so that can't happen, right ?

Wrong.

If you delete or deactivate a picklist value referenced in a process builder, even if the PB is active, Process builder will very kindly change the criteria to "select one". While staying active.

This affects ALL versions of the Process Builder, so you can't restore from an old one unless you stored the metadata somewhere. Seeing as Flows share the same backend, I would be wary about that as well.

All that to say Salesforce will, if your admins don't take care, kindly modify read-only automations, in prod, without notification. And of course if you undelete the value or undeactivate it, it doesn't bring it back.

Merry christmas.

PersonAccounts

Alright so PersonAccounts.
We've had issues with PersonAccounts at previous implementations. Some consultants lobbied to bring them back for B2C customers, feeling they deserved another chance.
That was a mistake. PersonAccounts are an ass, and we won't be working with them again.

All memes aside, here's a quick rundown of what PA is, when it's useful, and why most of the time it isn't.

WHAT ARE PA

To quote Salesforce "Person accounts store information about individual people by combining certain account and contact fields into a single record.".
Except that's wrong.

Technically speaking, Salesforce does not support a pure B2C model at all.
PersonAccounts were invented to fill that gap. Except how do you shoehorn B2C into what is an already sizable application at the time ?
- you could rewrite the entire thing to allow single contacts, which will also force you to rewrite your entire security model
- you could cheat and just merge two records together, call it a single record, and call it a day.

Option 2 is what Salesforce did.
So at the end of the day, a PersonAccount is just an Account and a Contact record, linked together via a couple lookup fields, that display as a single record in the UI.

This has a couple of interesting followups:

- PA have two IDs, one starting with 001 and one with 003
- PA take twice as much space as most SF records (so 4kb per record)
- PA can be used with most functions that are account or contact specific
- You can look from a record to a PA two times if you have an account and a contact lookup field

In general you can really just consider PA as the two individual records that comprise them, but masked to the user as a single one.

WHAT PA ARE SUPPOSEDLY USED FOR

PA are the answer to "I don't have accounts in my company" or "we're a B2C company". In theory at least.

PA work beautifully in small- to mid-size companies that have these considerations.
They're easy enough to set up, they answer that "no account" problem, and they're compatible with campaigns, hell they're even compatible with Marketing Cloud.

One thing that generally gets overlooked is that PA also lower the problem of having sharing settings for accounts, which is also great for those companies.

So if you have less than 500k customers, don't have tons of business-focused interactions, and want to do B2C, PA might be good for you.

WHY PA ARE ALMOST NEVER USED

The problem with the above are that PA are pretty much unfit for the purpose of B2C.

Before we get into bugs, let's look at that first paragraph again. It doubles your storage requirements. If you have 1 mil contacts, all of a sudden you're paying for 2 mil records in Salesforce. At big volumes, that is HUGE.

Then you have the sharing consideration.
"If you have enabled person accounts, the organization-wide default sharing must be set so that either Contact is Controlled by Parent or both Account and Contact are Private." That's a prerequisite, and you can't change it after activation.
So if you only have PA, you're fine.
But hell, if you have PA and normal accounts, you better hope you like sharing rules and apex sharing, because you're going to have a TON of it.

Let's add a sprinkle of dev in there.
Yes, PA are both Contact and Account. No, you can't use the Contact ExternalId fields to find them for some reason. Yes, querying the Account table will list PA. Yes, it will also show them in the Contact table. All of that is not blocking but it can make for hugely annoying situations.

Did I mention that Marketing Cloud supports PA but not great and that you'll almost certainly use Ampscript to compensate for it ?

And then there are the bugs. Process Builder doesn't ike PA at all. Hell even Workflows sometimes do weird shit if you're targeting email fields. APEX will have issues that devs are supposed to know as well.
It's just hell.

SO WHAT DO I DO WITH ACCOUNTS in B2C

Open your mind to the greatness of Cthulu.
Or otherwise, guide you clients to using accounts, not as companies, but as arbitrary groupings of contacts.
Hell a PA is just contact and Account anyway, so even if all contacts have a unique account, you're still at parity.

B2C commerce ? Use accounts as families! Base them on last name. Easy grouping of family members, nice reporting.
Insurance ? Do households ! Accounts are addresses with contacts in them You insure a place ! Yay !
None of that applies ? Why not use accounts as a billing entity ? Hell, whateer you do will be better than PA!

So TL;DR use them as arbitrary groupings of contacts that fit business needs.
Or, if you really, really fit the use case of small to mid-size company with no business processes that block PA and don't want to think too much about Architecture, use PA. And migrate away from them in a few years, because you're going to.

Love & Kisses!

Some Counterpoints and external links

https://www.quip.com/5XcaAbb43xWn/Person-Accounts-for-FSC-FAQs-Partners

https://developer.salesforce.com/docs/atlas.en-us.financial_services_cloud_admin_guide.meta/financial_services_cloud_admin_guide/fsc_admin_pers_impl_consid.htm

https://developer.salesforce.com/docs/atlas.en-us.financial_services_cloud_admin_guide.meta/financial_services_cloud_admin_guide/fsc_admin_data_model_diagram.htm

Case Assignment Rules triggering

Soooooooooo - Case Assignment Rules (I'll abbreviate this to CAR) don't actually trigger automatically and never do.

To trigger an assignment rule, a user must ALWAYS select "assign using active CAR".

The thing is, you can both hide this checkbox, and set it to TRUE by default.

So the user thinks they saved a case, but in effect they used a CAR MANUALLY.

What's fun is that when you load data, sometimes you need to process Assignment Rules. In the dataloader, this is done by specifying, manually, the assignment rule ID that you want to trigger. This can only be done by the Salesforce Data Loader as far as I know. Added fun !

So what does this have to do with your case ?

Turns out the ONLY way to trigger CAR on a case programmatically is with a trigger. You can't do it with a flow. You can't do it with a process builder. You just need to use DML.Opts.

A VERY simple trigger I copied from a Salesforce Board some time ago is below. Note that using this out of the box is bad. You want a dev to go over it and check your use case, as well as existing automations. Also you may notice that this is not best practice because everything is in a trigger. Short version this is an EXAMPLE and should NEVER be used in production.

```
trigger CaseTrigger on Case (after insert) {

Set<Id> caseIdSet = new Set<Id>();
for(Case c : trigger.new) {
caseIdSet.add(c.Id);
}

List<Case> caseList = new List<Case>();

Database.DMLOptions dmo = new Database.DMLOptions();
dmo.AssignmentRuleHeader.useDefaultRule = true;
```

```
for(Case c : [SELECT Id FROM Case WHERE Id IN: caseIdSet]) {
c.setOptions(dmo);
caseList.add(c);
}

update caseList;
}
```

anyway this will be the case for:

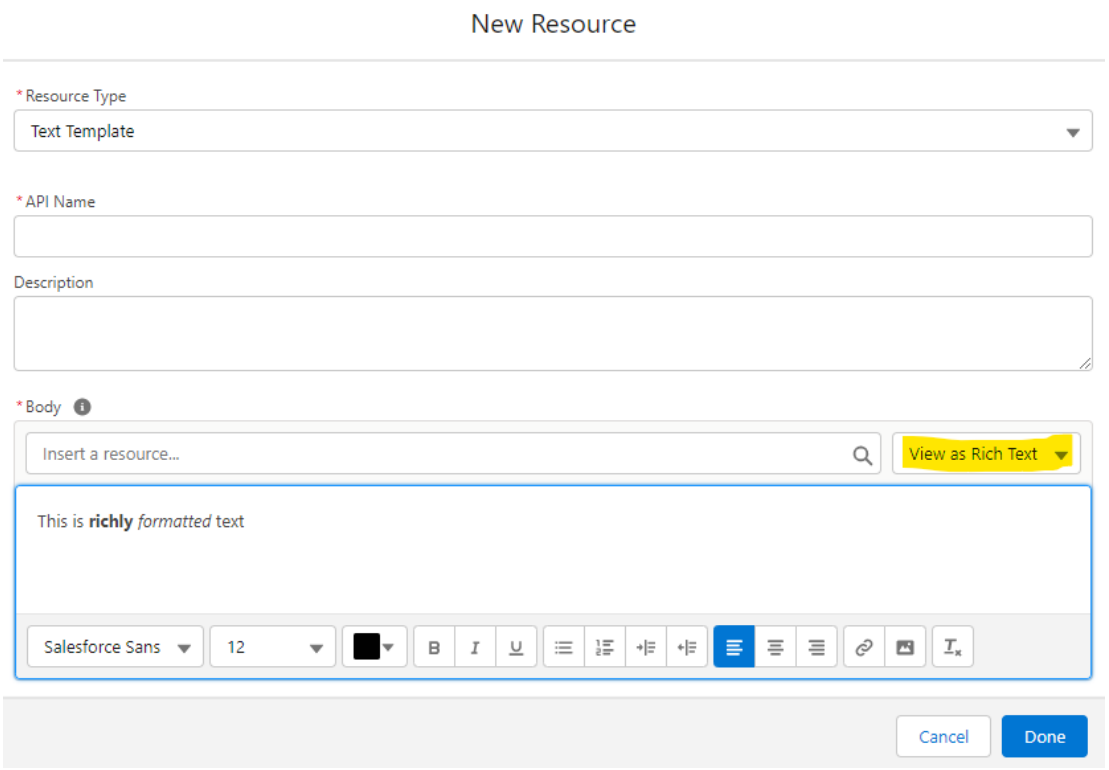
- social post cases

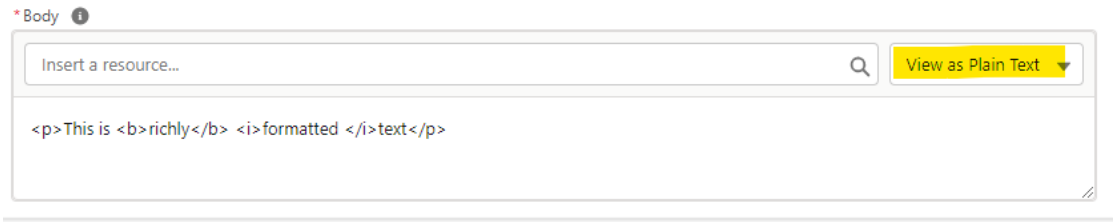
- community cases

- any whatever case that is created programatically.

Flow's Send Email Core Action + Text Templates

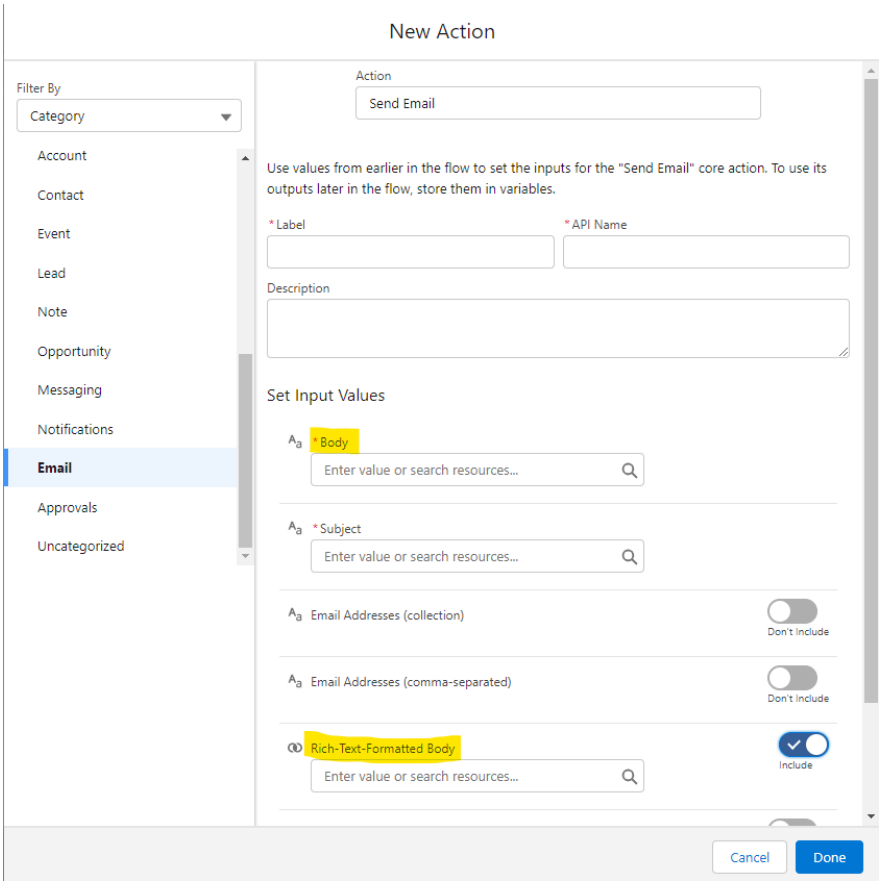
In Winter 20, Salesforce updated Flow Builder to make text template resources rich text as well as plain text. As such, if you have an existing Flow with a text template resource from pre-winter20' you will need to edit the Flow to ensure the proper type is chosen.





This is particularly an issue for using a text template as the source for the body of an email in a Send Email action. If you are using this text template when using an email core action, you need to ensure you have the Plain/Rich option selected correctly for whichever Body input you choose to use for the action:

- `Body` (default): If you use this default Input, and a text template as the source of it, that text template **must** be plain text or else it will render the rich text's markup directly in the body. To change this, select *Plain Text* from the drop down in the upper right.
- `Rich-Text-Formatted Body` (optional): alternatively, you can use a Text Template with Rich Text enabled and use this optional input to provide a formatted email body.



Permission Sets and Permission Set Groups: Shared values and silly rules

So, in experimenting with converting existing batches of Permission Sets into Permission Set Groups to ease the eyesore of having 100+ permission sets per user, I found the following interaction: Permission Sets and Permission Set Groups occupy the same developer space and as such, share unique developerNames. Nope, no 'Tier 1 Access' Permission Set and 'Tier 1 Access' permission set group.

In fact, Permission Set Groups are basically just a concatenated string of existing Permission Set Id's. They do not carry an actual recordId (if you export Permission Sets, the Permission Set Groups also get pulled in despite it being a 'separate' object; the recordId for Permission Set Groups just link you to the setup page for the Group; Permission Set Groups share the namespace prefix among Permission Sets as well.

Polymorphic Owners and field references

{!sobj_LoopedAccount.Owner.Email} is the right format but it returns null

when I put it on a screen it works fine

but in my scheduled flow it doesnt

ah nm I figured it out lol {!sobj_LoopedAccount.Owner.Email} is right.

BUT
{!sobj_LoopedAccount.Owner:User.Email} is what you want with polymorphic owners, like Cases
□□
Non-Pasta version - if you reference a polymorphic field in a scheduled flow, you MUST specifiy the type of relationship you are trying to follow - assuming that only one type of relationship will exist due to a pre-existing query will not work.
Even more succint: when following a polymorphic relationship, you should always specify which relationship type you are following.

That one Hardcoded Id

there is one ID in the world

that is ok to hardcode

```
recordTypeId: '012000000000000AAA'
```

the master picklist record id if your sobject has no record types

it's not very useful, except here

https://developer.salesforce.com/docs/component-library/documentation/lwc/lwc.reference_wire_adapters_picklist_values

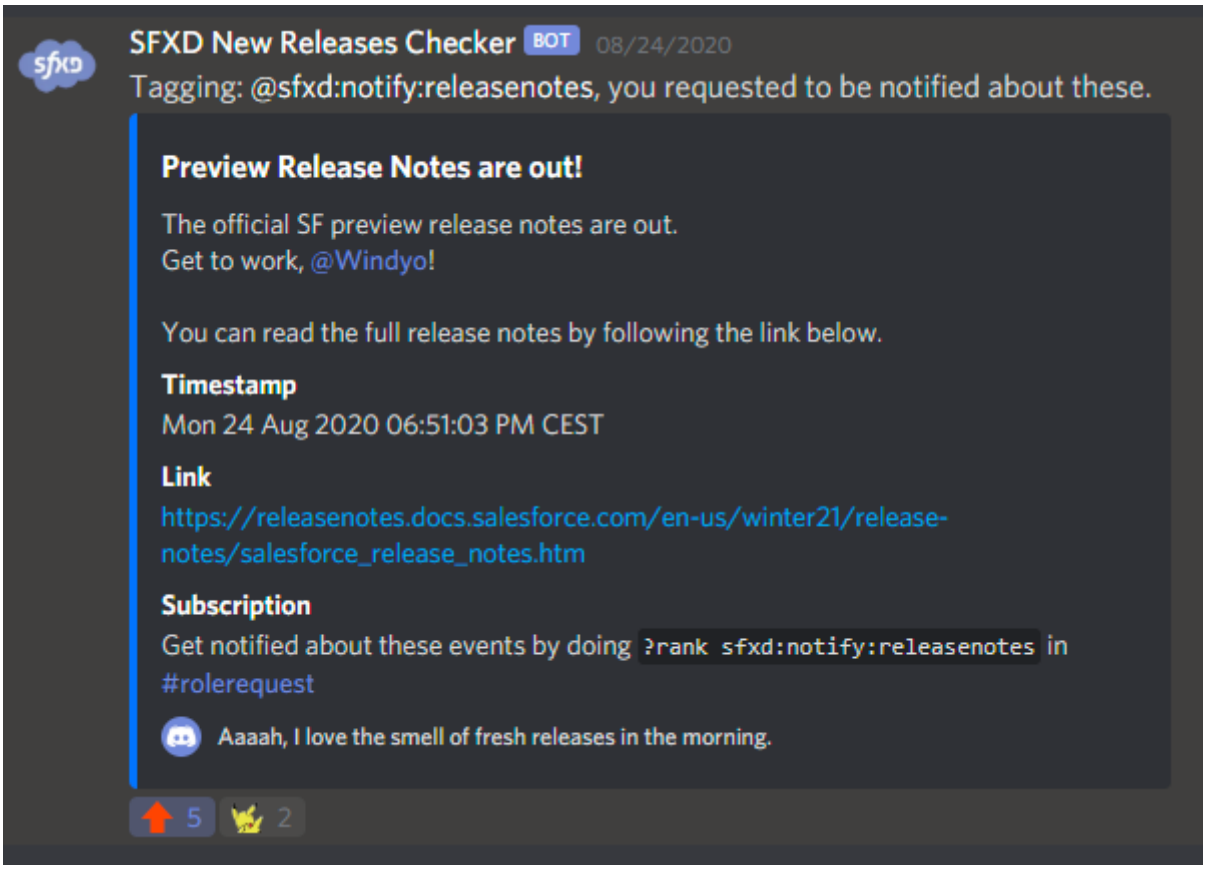
Screen_Shot_2019-09-12_at_1.52.36_PM.png

SFXD Presents - Parsing the new Help site is horrible

(Like really)
(Please don't break my parser)
(Plllllllllllllllllllllllllllllllleeeeeeeeeeeeeeeeeeeeeeeeease)

INTRODUCTION

One thing that I provide SFXD members (and myself, honestly), is a couple of bots.
Those bots do various things, but one of my favourites is my Notifications bot, which isn't really a bot, it's a shell script. A 10-line shell script.
This courageous little fellow, based on the current time of year as well as the actual year, constructs the next release (summer21, as of writing, or release 232), and then queries both the Release Notes page and the Preview Organization form - and if they go live (HTTP code 2XX), it pushes a nice notification in Discord. It also monitors the release notes for changes. All of this was easily done by querying the HTML of said pages.



Now obviously the script isn't perfect:

- Salesforce has a tendency to do weird things with its pages when nearing a release, making the pages come live and then not again, which results in bad notifications,
- using cURL to get the HTML means I was quite limited in processing unless I wanted to depend on pyton or another utility for parsing.

So I had been thinking of updating it to something more robust for a while.

Then, Salesforce decided it would move the Release Notes from
http://releasenotes.docs.salesforce.com/ to https://help.salesforce.com/.
The new website was slightly prettier, and didn't lose function, so I saw this mainly as an great excuse to finally change my notifier bot to something better.

... it didn't go so well.

THE RIGHT TOOL FOR THE JOB

So now that I had decided I wanted a better notification system I looked into various tools.

For those who know me, I'm somewhat of a self-hosting freak.
If I can self-host it, I will - it's funny, teaches me about server management, and that way I have the guarantee my tool won't disappear into the aether based on the whims of a company or

individual.

So I looked into Statuswatch, changedetection.io, and other such equivalents, before settling on Huginn. I, at the time, thought that it was WAY overkill but had the potential to be useful for other subprojects.
It turned out to be the PERFECT tool for the job, and I do not think this would have been doable without some heavy scripting if I had gone with another tool.

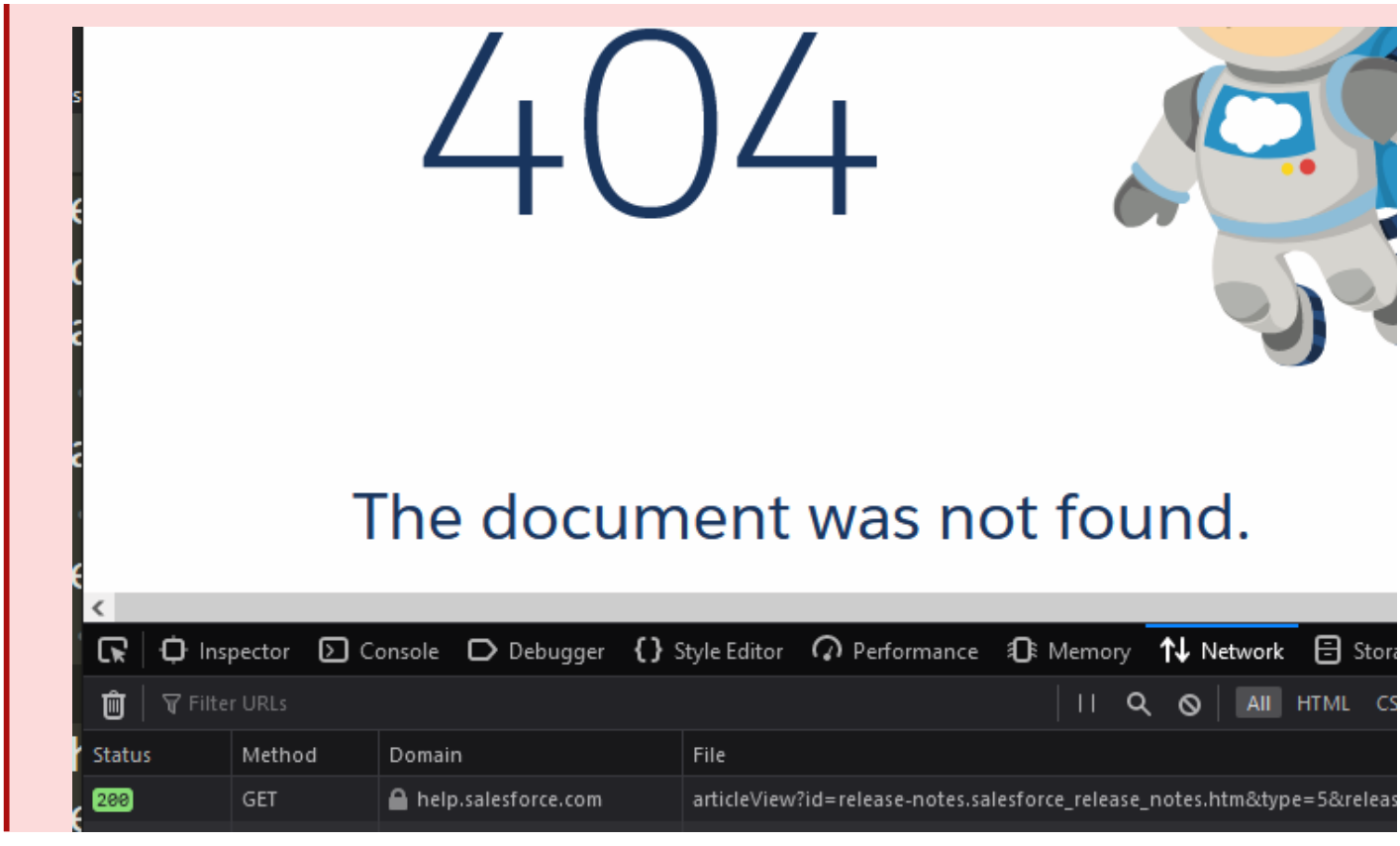
Yes, I am saying you need an entire events-stream based automation tool to parse release notes. We'll see why later.

GETTING THE RELEASE NOTES VERSION

In order to get the release notes, I need a few things:

- determine which release we currently are
- determine if a new version is coming
- determine if the webpage for the new version is up yet.

My original plan was to query `https://help.salesforce.com/articleView?id=release-notes.salesforce_release_notes.htm&type=5&release={{version}}`, much like I was doing previously, and trigger notifications if I had a confirmed (lasts more than a minute) `HTTP2XX` code - the old webpage returned a `302` if the version wasn't published yet.
A first problem arose: the Help site ALWAYS answers with `200`. Even if the version just doesn't exist at all. Try it: https://help.salesforce.com/articleView?id=release-notes.salesforce_release_notes.htm&type=5&release=96 the page *shows* you 404, but actually tells the browser "**YUP HERE'S YOUR PAGE.**"



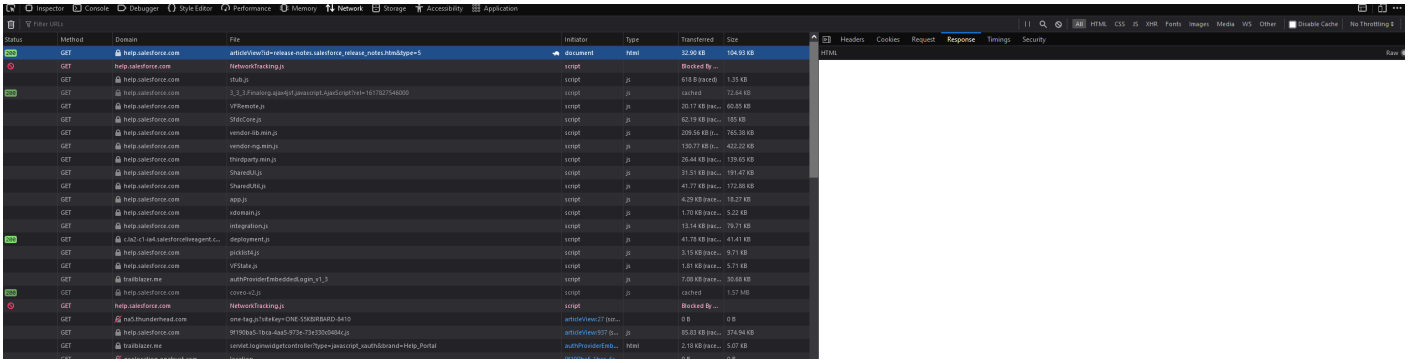
> Yes I know Help pages are mostly cost centers but this is just bad design

"No matter", I thought. "I'll just query https://help.salesforce.com/articleView?id=release-notes.salesforce_release_notes.htm&type=5, see which version number it returns, and check if `{{version +2}}` is a valid parameter". (version numbers increment 2 by 2 in Salesforce releases).
As you can guess, this doesn't work either. Funnily enough if you ask for the NEXT version, **Salesforce Help will still answer HTTP2XX in the rendered page**... but show you the current version. So 404 for invalid sessions, but 200 for next versions, OK, cool.

The next solution was then to get the current page, parse the version number, add 2, query that page, and check if versions differ.
And that's where the pain began.

GETTING THE RELEASE NOTES

Implementing the webpage fetcher was one of the most annoying things for this entire project. My original assumption was that I could just use a Huginn Website Agent (basically still a cURL command) to fetch the HTML, and then parse it using one of the many parsers Huginn offers. This would have worked if the Salesforce Help page worked like a normal web page. It does not.



such response, much wow

The Salesforce Help first answers any query with a wrapper, containing limited HTML and a LOT of Javascript but no other content.
This javascript then implements a Visualforce Remoting call, which queries Salesforce to display what seems to be KnowledgeArticles and Content on the page.
The result of those calls determine what you see afterwards.

The problem therefore was that Huginn, as well as cURL, sees this wrapper when doing their query. And they can't execute the JS calls themselves.

I fired up the Network inspector of Firefox, desperately looking for something that would avoid me having to fire up Selenium or something just as heavy-handed just to get the sweet text-based release notes I wanted.
LOTS of searching after, I found... nothing. Literally everything was handled through JS and fed

through VFremoting calls.
I naturally took the mature option and bitched about it to SFXD members, and @psb was kind enough to highlight that ONE call made on that page - the release notes Version picklist - was not to `apexremote` but to a static URI: `https://help.salesforce.com/services/apexrest/Help_TOCService?language=en_us&multiTocId=release-notes.salesforce_release_notes_toc&release={{version}}`

I hopefully queried it, and got back something I could work with: Raw JSON indicating the release number, the link to the PDF... and the HTML body embedded in an XML wrapper for some reason.

From there, the logic was simple:

- get `https://help.salesforce.com/services/apexrest/Help_TOCService?language=en_us&multiTocId=release-notes.salesforce_release_notes_toc`
- parse the version number, assuming it defaults to last stable release
- increment version number by two
- get `https://help.salesforce.com/services/apexrest/Help_TOCService?language=en_us&multiTocId=release-notes.salesforce_release_notes_toc&release={{stableversion+2}}`
- check if the release number in this subsequent call is equal to `{{stableversion+2}}`
- if yes, construct the release url `https://help.salesforce.com/articleView?id=release-notes.salesforce_release_notes.htm&type=5&release={{stableversion+2}}` send a notification with that URL, the PDF link, and the version number
- stop further notifications until the release number increments again.

This worked! (Except I can't know if it actually does until Salesforce releases the new preview so I can test it out, but at least the event stream worked.)

... but that wasn't the end.

GETTING THE RELEASE NOTE CHANGES

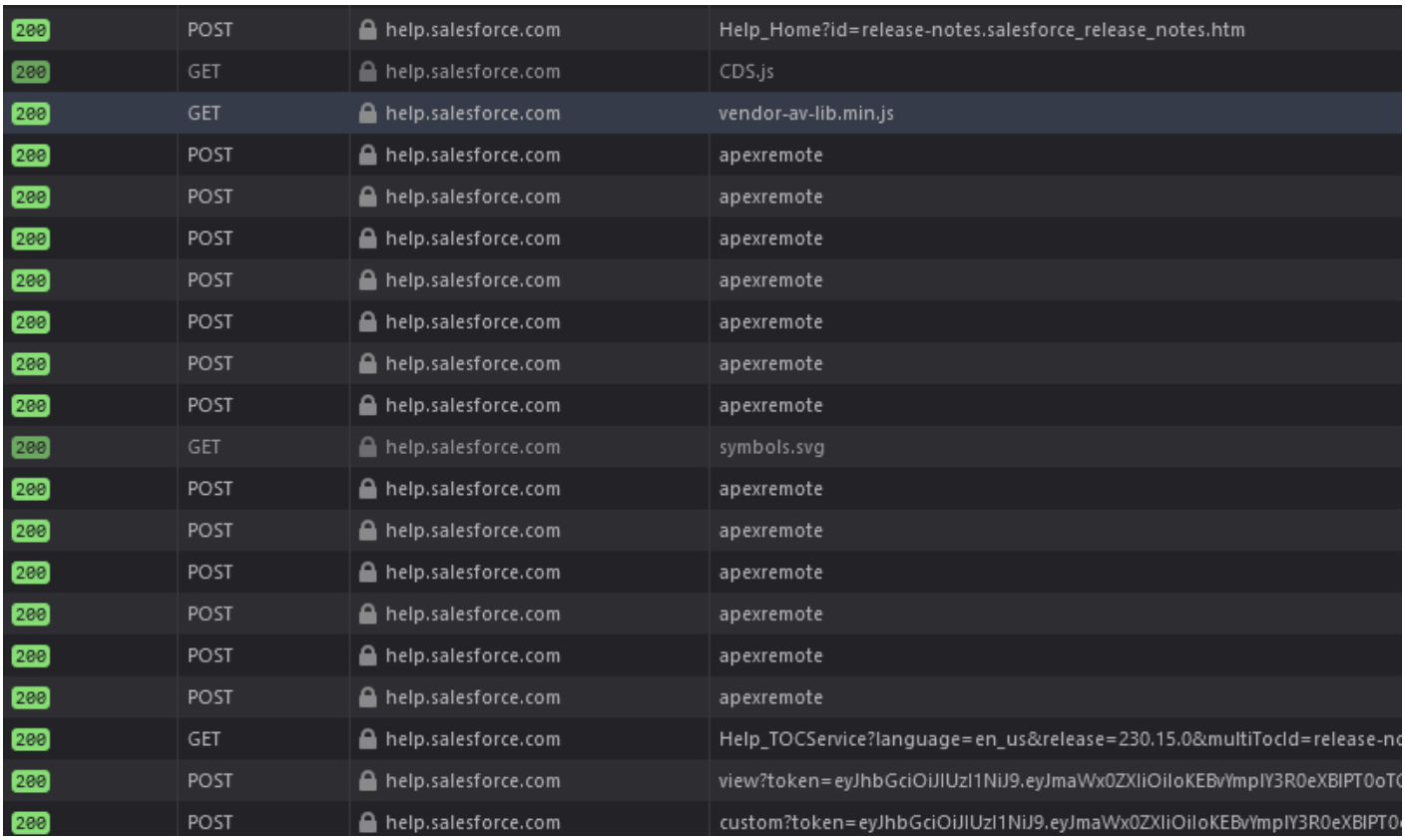
I had forgotten this part of it.
For those who don't know - the release notes, when not final, get modified. A LOT.

https://help.salesforce.com/articleView?id=release-notes.rn_change_log.htm&type=5&release=230
So for people like me who read them early, you gotta take the time to keep up to date. There's no schedule for these small updates, so using a change notifier was easier.

"*But you get the entire Release Notes body in some weird XML thing! You're fine!*" yes, that's what I thought. I hadn't checked said XML. Turns out it's just the headers for the top sections of the release notes, and I have no way of querying the changes to the release page.

So I went right back to the drawing board.

I was honestly pretty close to just paying for a PhantomJS cloud parser which could get all the page details for me. And then I decided "I'm already 6 hours in this project, it's 1AM, I might as well jsut go full ham". So instead, I decided to inspect every piece of information I got from every call, and try to get an APEX remoting call working via cURL.
After all, why not.



After checking all the calls done via `/apexremote/,` I found one called `getArticleData` which seemed to answer with the details I wanted in JSON format.

I used ReqBin to test a POST to `https://help.salesforce.com/apexremote`, using the same headers and payload as the browser. That yielded the same answer - cool. At least the authentication isn't a cookie.

Playing a bit with the headers and content made me realize a few things:

- I'll need to fake the Referer, Origin, and User Agent if I want `apexremote` to answer
- I'll need to send pretty much the same payload as the browser for it to resolve.
- This payload includes a VersionId, which I'm assuming I can get from somewhere... and a csrf token, because you wouldn't want people to browse your release notes from a non-Salesforce website, right. Gotta be secure.

The very first response I get from the Help Site as I said is mostly Javascript with no content. The JS yields some interesting information.

- one, there's a lot of parameters that seem to be some old stuff that just never got refactored (Hiiii org62 feedback button label)
- two, Visualforce.remoting.Manager is the class orchestrating everything and it seems to do some 143 calls before rendering the page. If you wonder why the original load is slow, that's why. Clicking on an article only does a VFremoting call for that one article - yay - but navigating to another menu item reloads everything. Which is sad considering it seems to load 135 of these elements on page load.
- three, the parameters for the VFRemoting manager are fed as inline JSON to this class. This inline JSON contains: the vfid, a list of actions including `getarticledata` AND a csrf token per action.

I had a way forward: I could now get that first page, parse the JS, parse the JSON within that JS, and then try to construct a POST that would pass verification based on the results therefrom. Not clean by any means, but what's a geek to do...

The good news is that getting said JS was easy enough. I just had to select all Script tags, emit those as individual events, and then filter on the one containing the Remoting Manager. I treated the resulting Javascript-which-contained-JSON as text, using regex to fetch the parameters the POST to `apexremote` needs. Dirty, but efficient.

The POST itself also went well. Turns out once you lie about where you are, which browser you are, which version you want and how the data is encoded, Salesforce answers happily to VF remoting calls done via scripts. I had a few issues with Huginn and its expectaiton that `application/json` be written simply `json` instead, which resulted in some weird parsing, but nothing major.

The data answered by the VF remoting call though is frankly horrible. It's a structured JSON, which contains a subobject which contains an XML wrapper for an HTML page which contains the changes to the release notes.
Due to some frustration with Huginn (mostly due to my lack of experience, but compounded by the fact that it did not allow me to view the raw data) I spent a LOT more time than I should have trying to parse this, but in the end I managed to get the holy grail - the html of the release notes changes page.

I then simply fed that to a change detector agent, which if it does detect changes, pings me on Discord.

TL;DR

The new SF help website isn't script-friendly, makes subscribing to changes REALLY hard, and its function is honestly weird - I do not see why we are hdiing release notes behind remoting calls protected via CSRF.
But well, I have a funcitonning notifier and change detector now.
Pic related

Agent Event Flow

Back