Hello, and thank you for joining me as I talk about some updates I’ve made to the Evergreen ILS Driver for VuFind. My name is Galen Charlton, and I am Implementation and IT Manager at the Equinox Open Library Initiative, a U.S. nonprofit organization that supports open source software for libraries.

This presentation will tell the story of how I updated the Evergreen ILS driver and what I learned along the way. While I will try to keep an eye on chat, I will plan on taking questions at the end of my presentation.
Background

• Why VuFind? ([https://vufind.org](https://vufind.org))
• Why Equinox? ([https://equinoxOLI.org/](https://equinoxOLI.org/))
• Why me?

So, what is Evergreen? Evergreen is an open source integrated library system that was originally developed for the Georgia Public Library Service’s public library consortium, PINES. PINES, which stands for Public Information Network for Electronic Service, currently serves about 300 libraries in the state of Georgia. The Evergreen Project was started in 2004 with the design goal of natively supporting the needs of library consortia. It had its 1.0 release in 2006. Since then, Evergreen has grown to encompass several large consortia in the United States and Canada with a presence in the Czech Republic. While Evergreen’s roots are in serving public library consortia, it has also been adopted by non-consortial public libraries and academic libraries. Evergreen had its release of version 3.8.0 just a few days ago.

Evergreen is, I would argue, without parallel for managing the sharing of physical resources among members of a consortium. Of course, libraries have many more electronic resources to manage than they did in 2004, and VuFind’s ability to index and provide access to electronic holdings from a variety of databases is a great complement to what Evergreen offers. Consequently, my employer, the Equinox Open Library Initiative, has started exploring VuFind and how we can use it to better serve our client libraries alongside our services for Evergreen, Koha, CORAL, and SubjectsPlus.

Why am I here? I have been involved in library automation for over 20 years, but in the past
15 years or so have been active in open source library software, primarily dealing with Koha and Evergreen. I’ve been an Evergreen project committer since 2010... and in the 11 years since then, have learned a thing or two about Evergreen. Consequently, when I started evaluating VuFind and its Evergreen ILS driver, I felt pretty comfortable with the Evergreen side of things. The VuFind side, however, was new to me.
“What Des-Cartes did was a good step. You have added much in several ways…”

Previous contributors to the Evergreen ILS driver include:

- Warren Layton, Natural Resources Canada (original author)
- Dan Scott
- Wesley Custer
- Demian Katz
- Josh Bannon
- Samuel Henrique
- Michael Birkner

But before I go on, first, an acknowledgment. The quote in the title of this slide comes from a letter that Isaac Newton wrote to Robert Hooke in 1675. It is followed by a sentence more commonly recognized: “If I have seen further it is by standing on the shoulders of giants”. In the same spirit, I want to acknowledge the work that many others have done on the Evergreen driver, including that done by Warren Layton, the original author, as well as Dan Scott and Wesley Custer, who I recognize as Evergreen users. But I also want to acknowledge the maintenance work done by many others: each commit updating the driver to reflect changes in API signatures, style guidelines, or internal conventions made by people who are not necessarily using Evergreen themselves has helped keep things fresh.
Entropy comes for us all

Much has changed in Evergreen since 2008:

• Updated password hashing algorithm
• Hold suspension
• DB schema changes

Of course, much has changed in VuFind... but the ILS driver interface has stayed surprisingly stable.

As near as I can tell, Warren Layton contributed the Evergreen ILS driver to VuFind in 2008, around the time that Evergreen 1.4 was released. Since then, both VuFind and Evergreen have evolved, though I must say that the VuFind ILS driver interface has been remarkable stable over these years.

On the Evergreen side, while most of the core ideas that Warren implemented remained the same, Evergreen has had several changes that resulted in the ILS driver getting increasingly out of date. The biggest change as far as the driver is concerned was in 2016, when Evergreen changed its password hashing algorithm from MD5 to one based on bcrypt.
What didn’t work?

- Patron authentication
- In some cases, retrieving the correct bib ID
- Displaying only active fines and hold requests

Because of that change, when I started looking at the driver, patron authentication simply didn’t work. It also quickly become apparent that the driver was returning Evergreen item record IDs in several places where VuFind was expecting title IDs. Fines and hold requests displays were also problematic.

To summarize: if you took a stock Evergreen system and a stock VuFind system, you could set it up to index records from Evergreen and retrieve real-time item status. But that was it: the Evergreen driver’s support for VuFind’s “My Research” or user account features just did not work.
What could be improved?

• Taking advantage of the loan display options
• Display of dates and times
• Display of owning and circulating locations
• Fetching more of the user profile

There was more that could be done in addition to fixing the core parts of the driver. I took advantage of the newer options that the VuFind ILS interface afforded to displaying current loans retrieved from the ILS. I also found that I could easily improve how VuFind displayed dates coming from Evergreen by hooking into VuFind’s built-in date and time formatters.

Evergreen, as a consortial ILS, makes a distinction between the library location that owns and item and the one that happened to circulate it, and I was able to add that to the getMyTransactions() response. Similarly, there was more that I could fold into the user profile response.
Let’s see some results

Note: no real patrons were harmed by having their library activity disclosed for the sake of a presentation slide.

Let’s take a look at the results. I should mention that what I’m about to show you is made-up data adapted from Evergreen’s test dataset; no real patron data is being shown today.
The result: user profile

The user profile was already mostly complete, but I did add the city, country and the patron’s privilege expiration date.
The list of checked out items had major problem: the driver was returning the item record ID, but VuFind wants the title or bib record ID. I’m not sure why the driver did this; all I can speculate is maybe that it was expecting a model where VuFind would have ingested a title record for each item record in Evergreen. Or maybe the driver was supposed to return the item ID back in 2008? In any event, I didn’t feel bound to that, so I changed the underlying query to return the Evergreen bib ID.

I also made changes to ensure that the loan statuses of due in the next 24 hours and overdue was reflected. Since Evergreen supports hourly loans, I also ensured that the due date string handed back to VuFind would show the time component if the loan was hourly. (Or technically speaking, if the loan duration was any interval not measured in whole days. If you want Evergreen to have a loan period of 7 minutes, you can do that, though I wouldn’t recommend inflicting that on your patrons.)
Similarly, `getMyHolds()` was returning the item ID, not the title ID. After fixing that, I also made changes to pass the frozen or thawed state of a hold request back to VuFind as well as its pickup shelf status.
As far as the fines interface was concerned... the Evergreen ILS driver started off as a clear fan of the no-fines movement.

Sadly, many libraries still charge overdue fines, and almost all libraries charge for lost items and photocopier fees and the like. Consequently, I corrected the getMyFines() method to correctly return data for both loan-related and non-loan-related (which Evergreen calls “grocery”) fines and fees.
Where are we at?

• VuFind can now be plausibly put in front of Evergreen
  • Not just for indexing and real-time status (as is the case of at least one production instance I’m aware of)
  • Now My Research / Patron Account features are available

So, where are we at? A good first step: VuFind is now ready to be plausibly put in front of an Evergreen system and provide both bibliographic access and patron transaction access.
Next steps for the current driver

- Finish putting in bind variables in the SQL statements
- Add some links back to the underlying Evergreen system
  - getCancelHoldLink, getHoldLink, and renewMyItemsLink
- Add some documentation to the wiki
- *Maybe* implement the course reserves retrieval methods

But... the current driver is a starting point, and can only be a starting point

Of course, there’s still more to do:

- Adding a bit more protection against SQL injection attacks
- Teaching VuFind how to direct patrons if they want to place holds or renewal items
- Adding some more documentation to the wiki. I’ve worked out some configuration changes needed to import and index MARC records from Evergreen, including how to map locations; I just need to write it up
- Maybe taking advantage of Evergreen’s recent course reserves features and add the course methods to the driver.

However, the current driver is *not* going to be a particular focus of mine (though I’m happy to fix bugs!). Let me show you some of the changes I made to it, which will help explain why a new direction is needed.
Here’s part of the change I made to getMyTransactions(). This is updating an SQL query to fetch data about the patron’s current loans from Evergreen... and that points to a problem with the current approach.

```php
$sql = "select circulation.target_copy as bib_id, " .
        "extract (year from circulation.due_date) as due_year, " .
        "extract (month from circulation.due_date) as due_month, " .
        '" due_day " .
+ $sql = "select call_number.record as bib_id, " .
+ "circulation.due_date as due_date, " .
+ "circulation.target_copy as item_id, " .
+ "aou_circ.name as borrowing_location, " .
+ "aou_own.name as owning_library, " .
+ "copybarcode as barcode " .
+ "from $this->dbName.action.circulation " .
+ "join $this->dbName.asset.copy ON (circulation.target_copy = copy.id) " .
+ "join $this->dbName.asset.call_number ON (copy.call_number = call_number.id) " .
+ "join $this->dbName.actor.org_unit aou_circ ON (circulation.circ_lib = aou_circ.id) " .
+ "join $this->dbName.actor.org_unit aou_own ON (call_number.owning_lib = aou_own.id) " .
+ "where circulation.usr = " .
+ "and circulation.checkin_time is null;" .
+ "and circulation.xact_finish is null;"
```
Problems with the current approach

• Direct DB connection
• Actually changing state via the DB is contraindicated
• Evergreen was designed for consortia and can have dozens or hundreds of libraries sharing an instance... not all of whom will necessarily agree about discovery interfaces

Namely, everything is being done via a direct database connection.

Now, SQL queries are fine – Evergreen uses a standard RDBMS, PostgreSQL, and is meant to be queried. But direct database connections are also risky: if you configure one, you’re putting a lot of trust in the client. Now, if you’re hosting VuFind and Evergreen on the same LAN, you can of course control both ends. However, sometimes you don’t control both ends. since one of Equinox’s services is software hosting, dealing with direct database connections over the internet is a situation we sometimes find ourselves in.

And... we hates it. Well, that’s too strong of a statement, but a raw database connection is a data security and patron privacy risk.

Furthermore, some of the potential features of VuFind would change state on behalf of the patron: placing hold requests, renewing loans, and so forth.

Could I successfully implement a routine to correctly place a hold request in Evergreen by direct database inserts and updates? Yes. Like I said, I’ve learned a thing or two from my decade working on Evergreen.

Do I want to? Should I? Would such a routine stay correct. Absolutely not!
Also, Evergreen helps bring libraries together to share their materials, but that doesn’t mean that members of a consortium necessarily all want the same discovery interface. Consequently, it’s important to be able to cleanly offer access to just a *subset* of an Evergreen system.
Mitigations

- Ensuring that the connection is encrypted
- Limited, read-only views
- *Scoped* read-only views

Now, there are some mitigations to help with securing a direct database connection. These include:

- Ensuring that the database connection is encrypted. This is easy in Evergreen’s case; PostgreSQL can both use and mandate that TLS be used.
- It’s also possible to have VuFind use a limited database account that’s read-only and has access to *only* the minimum set of tables and columns it needs.
- With a bit more effort, it’s also possible to provide limited access to just the rows “belonging” to a given library.

I’ll be writing some of this up for the VuFind wiki.
Next up: EvergreenAPI driver

- Will use the Evergreen API
- Data access will obey Evergreen’s permissions structure
- Changing state will become possible
  - Renewals
  - Hold requests
- Will allow consortia sharing an Evergreen system to share, or share not, VuFind systems as they prefer
- OAI-PMH for Evergreen?

However, the mitigations still don’t get us a way to change state on behalf of a patron.

Therefore, I will be working on a new Evergreen driver that will use Evergreen’s APIs. This will remove the need for a direct database connection and thus ensure that the data access respects Evergreen’s permission structure, both from the point of view of the patron using VuFind as well as the library or libraries providing access from Evergreen to VuFind.

Along the way, this will also give us a reason to help test a patch for Evergreen that’s been languishing for a while. That patch would allow Evergreen to act as an OAI-PMH provider, and if it works out, might be better than creating some custom export scripts to help keep a VuFind system up to date as its source Evergreen system gets updated.

I’m also expecting another side benefit for Evergreen. Evergreen web APIs conventions predate the development of a lot of standard expectations, e.g., that you’ll get responses in JSON, that data structures will be self-describing, and so forth. A good PHP client that uses Evergreen’s API would go a long way to help make it easier for others to integrate with Evergreen.
Might VuFind learn a couple tricks from Evergreen?

- Advanced hold requests
  - Evergreen does more than just title-levels and item-level holds
  - Include format?
- Metarecords?
  - E.g., place hold requests on an entire VuFind record version set?
- Fine payments?
  - Or at least, setting up a standard way to send patrons to a fine payment page
- Popularity metrics for search result ranking?

We’re exploring VuFind + Evergreen for a reason: VuFind’s search capabilities and ability to fold in non-ILS content would be a great complement to Evergreen.

But maybe there are a few things that VuFind (as a piece of software) might learn along the way as well.

For example, Evergreen not only lets you place title and item hold requests, but lets you place holds at the volume and metarecord (more on that soon) level. It also lets you place holds on a work and specify the formats that you care about without having to select a specific title. Making the Evergreen API driver be able to handle that would mean extending VuFind as well.

Similarly, Evergreen has a concept of metarecords that group together titles based on work keys. It’s a bit like VuFind record versions, except that Evergreen can do both searching and hold requests at the metarecord level.

Another area of interest are fine payments. In the short term, that probably means bouncing the patron over to the Evergreen fine payment page, but it might be interesting to explore adding payment gateway support to VuFind.
Evergreen also can include various measures of popularity (such as weighted circulation counts) to adjust relevance ranking and sorting. It’s not a widely used feature in Evergreen, but maybe there are some ideas worth exploring.
So, what was it like taking a reasonably large unit of work in a new-to-me codebase?

Pretty good, actually!

The ILS driver documentation was thorough (though I’m proud to say that I was able to spot and get fixed a minor glitch).

Feedback was just about instantaneous. I’d like to specifically call out and thanks Demian Katz for his responsiveness.

The continuous integration tests were easy to deal with.

And I was able to find just about everything I needed.

Oh, and installation of VuFind was pretty straightforward, although it did help that I’ve been working as a coder for quite a while. One thing I will mention, by the way, that I ended up using unionfs-fuse to set up a layered filesystem so that I could work on a Git clone at the bottom while letting things such as logs and Solr indexes live on the upper half, leaving the Git clone nice and clean. I’d be curious to know how others set up their development environments.
Using Evergreen+VuFind now

- Changes are in the ‘dev’ branch that will become VuFind 9
- [GitHub pull request](#) and [JIRA ticket](#)
- If you’re running VuFind 7.1 or 8.0, the changes should merge cleanly
- Is probably backportable to earlier versions of VuFind, but I can’t guarantee it
- I’ll be adding some more instructions to the wiki

So, suppose you’re running Evergreen now and want to try VuFind?

My changes are in the ‘dev’ Git branch (thanks Demian!) and will be part of VuFind 9. However, they should be readily backportable to VuFind 7.1 and 8. I suspect that they can be backported to earlier versions of VuFind without much fuss, although more work would be required to get it working before VuFind 5.1. However, I’m not making any guarantees here.

MARC import into VuFind from an Evergreen export is straightforward, but I’ll be writing up some instructions for the wiki.
Onward

- Working to establish some libraries using both VuFind and Evergreen
- Open invitation: there will be an online Evergreen conference in 2021. VuFind folks are welcome!

It has been a pleasure working to improve the Evergreen driver for VuFind. We at Equinox are aiming to establish some more libraries using both VuFind and Evergreen, joining the one production instance that I’m aware of. As a member of the Evergreen Project, I’d also like to extend an invitation: please join us at the online Evergreen Conference in 2022. I’ll share details as they become available.
Thanks!

Galen Charlton  
Implementation and IT Manager  
Equinox Open Library Initiative  
gmc@equinoxOLI.org

Thank you for your time and attention! As I am literally incapable of building a slide deck that does not include pictures of my cats, please enjoy this one of my orange tabby George cuddling up to Lucifer.

If you have any questions or feedback for me, please go ahead!