2023-07-28

Slow progress on app save/restore support

25 minutes allocated. 20 minutes for talk, 5 minutes for questions.

Hello. Today I'm going to talk about where we're at with app save/restore support, the architecture being proposed for it, and what's left to do. I should probably stifle anyone's hopes right now: this is not a big reveal talk in which I show that it's actually all complete and ready for app authors to use. It's a long way from that, but hopefully where we're at now is interesting enough to share.

2023-07-28

Slow progress on app save/restore support

This project is something I've been working on in periodic free hacking time at Endless, for a couple of years now. It's based on work done by Bastien Nocera a few years ago, and incorporating ideas from Emmanuele Bassi. As far back as 10 years ago, people were having discussions and ideas about how to implement this, so there's a long history of work this builds on. On the GTK side of things, Matthias Clasen has done some experiments in the last couple of years with widget tree serialisation/deserialisation, with the aim of helping implement this feature.

Slow progress on app save/restore support

What is app save/restore support?

What is app save/restore support?

- Reopening an app in the same state as it was when it was closed
- Closing by user action or by system policy
- gnome-session used to support a limited version of this in the olden days

Firstly, what is app save/restore support? It is what it says on the tin: restoring an app to the same state it was in when it was closed, when it's next opened. For example, with the same windows showing, in the same position, with the same list items selected and the same scroll position in views.

Slow progress on app save/restore support

What is app save/restore support?

What is app save/restore support?

- Reopening an app in the same state as it was when it was closed
- Closing by user action or by system policy
- gnome-session used to support a limited version of this in the olden days

This could happen either when you log out and log in again, or when the system decides to free up resources by killing a process and then transparently restarting it when you switch focus to it again. Android does this, but we're a long way off supporting it.

Slow progress on app save/restore support

What is app save/restore support?

What is app save/restore support?

- Reopening an app in the same state as it was when it was closed
- Closing by user action or by system policy
- gnome-session used to support a limited version of this in the olden days

gnome-session used to support a very limited version of app save/restore, in that it would save the list of apps you had open when logging out, and start those apps again when you next logged in. The apps would start from fresh, though, with none of their internal state saved. This was supported (I believe) until GNOME 3. In particular, when `systemd --user` support was added to gnome-session, the save/restore support broke.

Slow progress on app save/restore support

Use cases

Use cases

- Restore apps to largely the same state when logging out and in again
- Restore apps to largely the same state if they are killed and later restarted by the session manager due to resource constaints
- Restore apps to largely the same state if they are updated and restarted due to an update (e.g. flatpaks)
- Allowing apps to choose *not* to be restored (even if the data is available) if it would not be appropriate

So I've been working on parts of app save/restore on and off for a couple of years now, in various periods of hacking time provided by my employer, Endless. The use cases I've had in mind are these (I won't read them out).

Slow progress on app save/restore support

Use cases

Use cases

- **Not**: Allowing apps to checkpoint their state or provide an undo stack
- **Not**: Allowing apps to be restored if they are killed unexpectedly, crash, or power is lost

There are also a few things which are explicitly not use cases for this feature. This is to prevent it ballooning into something which can track arbitrary app state over time.

Slow progress on app save/restore support

Proposed architecture

Proposed architecture

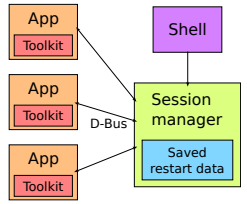

Figure: Proposed system architecture

Overall, it should actually be pretty simple to implement save/restore support in a single app. It actually only takes changes in the app itself to serialise and deserialise its state to a file on start and exit. The thing is, in order to have the app correctly restarted at the start of the session, you need help from the session manager. And it needs to have a standard interface to talk to, so you need help from the toolkit. And it would be helpful for the toolkit to provide a standard interface for serialising/deserialising a widget hierarchy. And then in order to implement the use case for closing apps when the system is low on memory, you need to know which app is focused, so the Shell becomes involved. So before you know it, the project involves four different components (or five if you count GTK and GLib separately).

Slow progress on app save/restore support

Proposed architecture

Proposed architecture

- Extend the session manager protocol
- Apps get their restore data when they register with the session manager
- Save their data when the session ends (if they're still running)
- Expose their data on `org.gtk.Application` as well, so it can be queried any time
- Data is an arbitrary `GVariant`

For the moment, let's ignore anything to do with the shell, as that's not something I've looked at yet. With that out of the way, the core of the proposed architecture is some changes to the session management D-Bus API. This is used for apps to communicate with gnome-session, so it knows what processes are running in the user's login session. It uses this to move them to the right cgroup, to tell them when the session is going to end (and to allow them to inhibit logout/shutdown), and now to get their save/restore data from them.

Slow progress on app save/restore support

Proposed architecture: restoring

Proposed architecture: restoring

Figure: Registering a restartable client on session start

This is the architecture I've ended up with for restoring app state, which happens on startup. gnome-session will auto-start a list of apps saved from the last session. When each of them register with the session (historically via the `RegisterClient` method), the session manager will now send their restart data back to them. The app can then deserialise this to restore their state.

Slow progress on app save/restore support

Proposed architecture: restoring

Proposed architecture: restoring

Figure: Registering a restartable client on session start

This fits in well because the session management API calls happen before the app shows any UI, and happen already, so this approach adds no additional D-Bus traffic. Restore data can be loaded from disk once by the session manager, rather than N times by N apps.

Slow progress on app save/restore support

Proposed architecture: restoring

Proposed architecture: restoring

Figure: Registering a restartable client on session start

The diagram shows a Bustle trace of the D-Bus traffic for the proposed API. In it, the session manager auto-starts an app, the app registers with the session manager and receives its restart data, and then its normal startup process continues.

Slow progress on app save/restore support

Proposed architecture: saving

Proposed architecture: saving

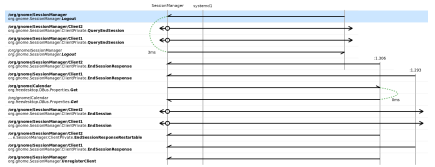

Figure: Saving app state when ending the session

The architecture for saving app state is similar. It happens when the session ends (when the user logs out, shuts down, restarts). The session manager signals all apps querying whether they're OK with the session ending. At this point they have an opportunity to inhibit the session ending, for example because they have unsaved documents. Once all documents are saved, the session manager notifies all apps that the session is ending, and they respond with `EndSessionResponseRestartable`, which acknowledges the session ending, and contains their serialised restart data, or a boolean saying they explicitly don't want to be restarted on next session start.

Slow progress on app save/restore support

Proposed architecture: saving

Proposed architecture: saving

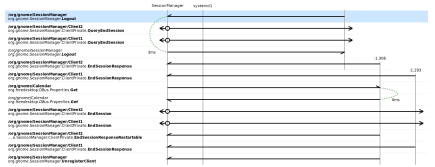

Figure: Saving app state when ending the session

The `EndSessionResponseRestartable` call is where this protocol differs from before, as the restart data has been added to it. As with restoring an app, this protocol change introduces no new D-Bus calls, and happens after the app has been notified of session end — so it has opportunity to prevent the user from interacting with the UI further, and hence there are no races when saving app state.

Slow progress on app save/restore support

Application interface

Application interface

```
struct _GApplicationClass
{
  GVariant *(* build_restart_data) (GApplication
*application,
                                    char
**out_tag);

  void      (* consume_restart_data)(GApplication
*application,
                                     const char
*tag,
                                     GVariant
*data);
};
```

The interface between application code and the toolkit is this pair of new virtual methods on `GApplicationClass`. The intention is that they could either be implemented directly by apps (if they want to do their own serialisation) or chained up to GTK (to use its serialisation). In either case, the app would probably have to explicitly opt in to restart support.

Slow progress on app save/restore support

What's implemented?

What's implemented?

- Application interface is ready for review in GLib
- Session manager API changes ready for initial review
- GTK session management changes are the right shape
- GTK widget tree serialisation/deserialisation not done
- Shell integration not thought about

What's implemented is basically what I've run through so far. I have enough changes locally to be able to run `gnome-session` and have it start `gnome-calendar` with saved data, then query the calendar for data at the end of the session and save that. I'm not brave enough to try and demo that in this talk, so you'll just have to take my word for it! It wouldn't be very exciting anyway, because the exciting visual part would come from the GTK widget tree serialisation/deserialisation, and none of that is implemented yet.

Slow progress on app save/restore support

Current problem points

Current problem points

- Little thought given to non-GNOME desktop, non-GTK apps
- Little thought given to XSMP apps
- Prototype for GTK serialisation is a few years old and I haven't incorporated it
- API has to be stable, so it has to be right

Of the things which have been written so far, there are a few areas which I think could do with some more thought, which I haven't done yet. Eventually, this should work cross-desktop, cross-toolkit, and degrade nicely for old apps which don't support it (even the really old apps which speak XSMP rather than the D-Bus gnome-session protocol we've used for the last decade plus). So validating the API changes from these points of view would be useful.

Slow progress on app save/restore support

What's next?

What's next?

- Help to validate the architecture
- Come to the GTK BoF session (Saturday, 10:00)!
- Landing some initial bits
- gnome-session changes
- Future work on GTK and Shell

What's next? I need your help! If you're interested in this and can contribute to any of the necessary components, please come to the GTK BoF session on Saturday. We'll be discussing app save/restore as part of the session there.

Slow progress on app save/restore support

What's next?

What's next?

- Help to validate the architecture
- Come to the GTK BoF session (Saturday, 10:00)!
- Landing some initial bits
- gnome-session changes
- Future work on GTK and Shell

In terms of landing things, the GLib changes are ready for review and hopefully to be landed this cycle. The gnome-session changes are ready for review, but I expect there will be a lot of comments on them. Particularly, I'm interested in high-level comments on the D-Bus API changes at the moment. The GTK changes at the moment are quite minimal and tightly paired to the gnome-session changes. Perhaps a good way to land the gnome-session changes would be to gate them behind a configuration option to begin with, then they could land earlier and get more testing.

Slow progress on app save/restore support

What's next?

- Help to validate the architecture
- Come to the GTK BoF session (Saturday, 10:00)!
- Landing some initial bits
- gnome-session changes
- Future work on GTK and Shell

There will need to be a lot of separate work done on serialising/deserialising widget trees in GTK to make that convenience available for apps. And I haven't looked at the Shell changes at all, which is what's necessary for killing apps when system resources become scarce.