[PUBLIC DOC]

- BrowserCollection
  - Should be platform-agnostic
  - Should be able to rename browser_collection_non_android.cc back to browser_collection.cc and use it on both platforms
- Options for GlobalBrowserCollection and ProfileBrowserCollection
  - Inheritance
    - Pros
      - Subclass automatically gets access to (protected) superclass members
    - Cons
      - Will need separate per-platform headers
      - Static methods will need to be manually delegated anyway
  - Delegation
    - Pros
      - May be able to have shared headers between platforms for ProfileBrowserCollection
    - Cons
      - Need to add delegate as friend, pass in superclass
- Who gets observed?
  - Platform-agnostic class, or delegate?
  - Answer: top-level class gets observed, not delegate. Observing the delegate would be cumbersome and need lots of code changes at clients. Delegate can be friend class and effectively act as if it's just part of the same class.
- Can GlobalBrowserCollection still be a GlobalFeature on Android? Startup ordering may need investigation.
- See https://docs.google.com/document/d/12un51kAcjvZ6fVg1IhxOIaDKkdbgCc9LJHIx1ZQDk98/edit?tab=t.0 for Profile-related issues to keep in mind (Google-internal, sorry)

```
C/C++
// browser_collection.h - unchanged (https://source.chromium.org/chromium/chromium/src/+/main:chrome/browser/ui/browser_window/public/browser_collection.h) as there are no platform-specific components in either the header or the implementation file. The implementation file will be merged/renamed back to browser_collection.h.
```

C/C++

```
// global_browswer_collection.h - platform-agnostic, implemented in
global_browser_collection.cc

class GlobalBrowserCollection : public BrowserCollection {
 public:
  GlobalBrowserCollection();
  GlobalBrowserCollection(const GlobalBrowserCollection&) = delete;
  GlobalBrowserCollection& operator=(const GlobalBrowserCollection&) = delete;
  ~GlobalBrowserCollection() override;

  // ideally platform-agnostic, depending on whether we can still just be a
GlobalFeature on Android
  static GlobalBrowserCollection* GetInstance();

  // BrowserCollection:
  bool IsEmpty() const override; // platform-agnostic
  size_t GetSize() const override; // platform-agnostic

  GlobalBrowserCollectionPlatformDelegate* GetPlatformDelegate(); //
platform-agnostic, used by (for example) BrowserManagerService for observation on
desktop

 protected:
  // BrowserCollection:
  BrowserVector GetBrowsers(Order order) override; // platform-agnostic

 private:
  friend base::ScopedObservationTraits<GlobalBrowserCollection,
                                       BrowserCollectionObserver>;
  friend GlobalBrowserCollectionPlatformDelegate;

  GlobalBrowserCollectionPlatformDelegate platform_delegate_; // has different
implementations per-platform

  // likely can be shared between platforms here, as opposed to being in the
platform-specific delegates. Not fully confirmed.
  std::vector<raw_ptr<BrowserWindowInterface>> browsers_creation_order_;
  std::vector<raw_ptr<BrowserWindowInterface>> browsers_activation_order_;
};
```

C/C++

```
// global_browser_collection_platform_delegate_android.h - Android-specific,
implemented in global_browser_collection_platform_delegate_android.cc

// may need a Java impl so it can be an instance of
https://source.chromium.org/chromium/chromium/src/+/main:chrome/browser/ui/browser_window/public/android/java/src/org/chromium/chrome/browser/ui/browser_window/ChromeAndroidTaskTrackerObserver.java (similar to
https://source.chromium.org/chromium/chromium/src/+/main:chrome/browser/ui/browser_window/internal/android/java/src/org/chromium/chrome/browser/ui/browser_window/AndroidBrowserWindowEnumerator.java)
class GlobalBrowserCollectionPlatformDelegate {
 public:
  explicit GlobalBrowserCollectionPlatformDelegate(
      raw_ref<GlobalBrowserCollection> parent);
  GlobalBrowserCollectionPlatformDelegate(
      const GlobalBrowserCollectionPlatformDelegate&) = delete;
  GlobalBrowserCollectionPlatformDelegate& operator=(
      const GlobalBrowserCollectionPlatformDelegate&) = delete;
  ~GlobalBrowserCollectionPlatformDelegate();

  // other Android implementation details may go here

 private:
  // notice the lack of BrowserCollectionObserver implementation - that is only
relevant on desktop.

  raw_ref<GlobalBrowserCollection> parent_;
};
```

C/C++

```
// global_browser_collection_platform_delegate_non_android.h -
Non-Android-specific, implemented in
global_browser_collection_platform_delegate_non_android.cc

class GlobalBrowserCollectionPlatformDelegate
    : public BrowserCollectionObserver {
 public:
  explicit GlobalBrowserCollectionPlatformDelegate(
      raw_ref<GlobalBrowserCollection> parent);
  GlobalBrowserCollectionPlatformDelegate(
      const GlobalBrowserCollectionPlatformDelegate&) = delete;
  GlobalBrowserCollectionPlatformDelegate& operator=(
```

```
      const GlobalBrowserCollectionPlatformDelegate&) = delete;
  ~GlobalBrowserCollectionPlatformDelegate() override;

 private:
  // BrowserCollectionObserver:
  void OnBrowserCreated(BrowserWindowInterface* browser) override;
  void OnBrowserClosed(BrowserWindowInterface* browser) override;
  void OnBrowserActivated(BrowserWindowInterface* browser) override;
  void OnBrowserDeactivated(BrowserWindowInterface* browser) override;

  raw_ref<GlobalBrowserCollection> parent_;
};
```

C/C++

```
// global_browser_collection_platform_delegate.h - for convenience so includers
don't need to think about platforms or ifdefs

#if BUILDFLAG(IS_ANDROID)
#include "global_browser_collection_platform_delegate_android.h"
#else
#include "global_browser_collection_platform_delegate_non_android.h"
#endif
```

C/C++

```
// profile_browser_collection.h - platform-agnostic, implemented in
profile_browser_collection.cc

class ProfileBrowserCollection : public BrowserCollection {
 public:
  explicit ProfileBrowserCollection(Profile* profile);
  ~ProfileBrowserCollection() override;

  static ProfileBrowserCollection* GetForProfile(Profile* profile); // delegated

 private:
  // This member may actually end up being unnecessary, because the
platform-specific parts of the Android implementation may be implemented in a
separate Android-specific keyed service, like BrowserManagerService for desktop.
```

```
The only need for a "delegate" in the Profile case may be the static
ProfileBrowserCollection getter.
  ProfileBrowserCollectionPlatformDelegate platform_delegate_;

  const raw_ref<Profile> profile_;

  friend base::ScopedObservationTraits<ProfileBrowserCollection,
                                       BrowserCollectionObserver>;
  friend ProfileBrowserCollectionPlatformDelegate;
};
```

```
C/C++
// profile_browser_collection_platform_delegate.h - (ideally) platform-agnostic
header, but separate platform implementations in
profile_browser_collection_platform_delegate_android.cc and
profile_browser_collection_platform_delegate_non_android.cc. The main logic for
the Android implementation will likely go in a new Android-specific keyed service
(like BrowserManagerService for desktop) that inherits from this class and is also
a BrowserCollectionObserver to observe the GlobalBrowserCollection.
// OR: could make the Android implementation of this class into an observer
directly (like GlobalBrowserCollection desktop delegate)

class ProfileBrowserCollectionPlatformDelegate {
 public:
  explicit ProfileBrowserCollectionPlatformDelegate(
      raw_ref<ProfileBrowserCollection> parent);
  ~ProfileBrowserCollectionPlatformDelegate();

  static ProfileBrowserCollection* GetProfileBrowserCollectionForProfile(
      Profile* profile);

 private:
  raw_ref<ProfileBrowserCollection> parent_;
};
```

Tests

- Browser_collection_unittest.cc does not currently compile on Android, because mock_browser_window_interface.h makes non-Android assumptions. We'll need to add ifdefs to mirror browser_window_interface.h.

- ProfileBrowserCollection and GlobalBrowserCollection browsertests not known yet, but suspect they'll need work to run on Android (may need separate per-platform impls for delegates)

Plan

- Proceed with BrowserCollection and GlobalBrowserCollection parts for now (including tests). WIP impls (tests not ready yet, and GlobalBrowserCollection::GetInstance() ifdef'd for now): https://chromium-review.googlesource.com/c/chromium/src/+/7485129, https://chromium-review.googlesource.com/c/chromium/src/+/7485130, respectively.
- Hold off on ProfileBrowserCollection until it's more clear whether we'll need 2 separate profile_browser_collection_platform_delegate headers (eg. if the Android impl needs to be a BrowserCollectionObserver, what its keyed service will look like, etc). (FWIW, here's a sample impl, but it makes some assumptions we might not want to land yet:https://chromium-review.googlesource.com/c/chromium/src/+/7485131)

Further discussion

Because the `global_browser_collection_platform_delegate.h` header which just conditionally includes other headers is contentious.

Problem: using ifdefs in a header just to include other headers is gross/sketchy/weird/unusual/magic/etc. Arguably having the interface definition of the `PlatformDelegate` being different on different platforms could lead to confusion (having the implementation differ is one thing, but maybe the interface is another level). Maybe there's a more idiomatic way to do it.

Options:

1. The original proposal, as above
    - One type name, but different types on different platforms (both header and impl)
    - The main nice feature here is the lack of ifdefs. The only ifdefs are in `global_browser_collection_platform_delegate.h`, whose sole purpose is the ifdefs. Every other file is either 100% Android, 100% non-Android, or 100% platform-agnostic. (Ignore the ifdef in `GlobalBrowserCollection::GetInstance()` - it's just there temporarily to make it compile until I figure out if `GlobalBrowserCollection` can be a `GlobalFeature` on Android).
    - One concern is that we're "pretending" that there's a single `GlobalBrowserCollectionPlatformDelegate` type, whereas in reality they are different on different platforms. I argue that this isn't functionally different from having inline ifdefs like we have all over the place in `browser_window_interface.h`. Splitting it into separate files risks having

more repeated code (in cases like BWI), but makes it much easier to read IMO - and in this case we don't really expect any repeated code, because the repeated parts will be in `GlobalBrowserCollection`, not in the delegates.
    - Overall generally just "makes sense" to me. The platform-specific parts are each separated into their own place, the platform-agnostic parts don't need to know that there are different platforms, no ifdef messiness. It seems the most clean and readable to me. I could see the argument that separate platform implementations is unexpected and confusing in some cases, but here the name `PlatformDelegate` should imply platform-specificness.
    - Precedent: tabs_event_router.h (ok, technically this is an example of option 2, but they're very similar)
    - Prototype: 7485130
2. The proposal above, except using ifdefs at include sites instead of a separate header dedicated to just ifdefs
    - Pretty much the same thing, but maybe it helps readers understand more up-front that there are per-platform differences
    - One fewer file needed
    - Arguably not very nice having to use ifdefs in a file that's supposed to be platform-agnostic
    - Precedent: tabs_event_router.h
    - Prototype: 7486517
3. Let GN decide which header to include
    - Not sure if this is possible, and if it is, it sounds like it's uncommon / unidiomatic
4. Have a single cross-platform `GlobalBrowserCollectionPlatformDelegate`, then have separate per-platform `AndroidGlobalBrowserCollectionPlatformDelegate` and `NonAndroidGlobalBrowserCollectionPlatformDelegate` subclasses
    - Doesn't avoid ifdefs (may, in fact, introduce extra ifdefs)
    - `GlobalBrowserCollection` needs to 'friend' both platform impls
    - It is nice that we explicitly have different types per platform, rather than pretending we have one `GlobalBrowserCollectionPlatformDelegate` which is actually 2 different things depending on the build config. This may make things like stack traces and crash dumps easier to reason about? (Though those typically also contain the filename anyway).
    - Precedent: system_signals/base_platform_delegate.h
    - Prototype: 7488761
5. Include both header file implementations in all builds (using different class names), but conditionally typedef per platform
    - Not sure of many concrete advantages, other than possibly a feeling that it's nicer than the conditional header hackery (and, I suppose, you can put both impls in a single header rather than using 2-3 files)
    - Perhaps has the same stack trace / crash dump advantages as option 4, though?

    - Doesn't work if there's anything platform-specific in the header (ie, references to types that only exist on one platform, #includes of other headers that are only supposed to be used on one platform, etc) unless we also put ifdefs all over the place
    - Precedent: native_ui_types.h, platform_browser_test.h (sort of)
    - Prototype: 7486833
6. Make the desktop delegate contain an observer as a member, rather than being an observer itself (ie, a single shared cross-platform header for the delegate). The Android impl has a nullptr observer.
    - Kind of feels like having a delegate for the delegate
    - Makes BrowserManagerService slightly uglier (we'd have to do `GlobalBrowserCollection::GetInstance()->GetPlatformDelegate()->GetObserver()` or similar - `GlobalBrowserCollection::GetInstance()->GetPlatformDelegate()` is already not ideal). We could work around this with a separate static, like `GlobalBrowserCollection::GetObserverInstance()`.
    - In order for observer to be able to access the browser lists, it, too, needs to be a friend of GlobalBrowserCollection
    - Precedent: [TODO - not sure I'll be able to find any, even if they exist]
    - Prototype: 7485102
7. Have 2 completely separate types with separate names (no inheritance) and just ifdef them in the `global_browser_collection.h` header
    - Basically option 1 again, but have explicitly different names for the different platforms, plus some extra ifdefing
        - different names for the different platforms -> again maybe better for stack traces
    - At the same time, it's also basically option 5, but with more ifdefs instead of a typedef (except we can have platform-specific stuff in the headers, because they're only included on their own platform)
    - Precedent: [TODO - haven't looked too hard yet, but I'm sure this must exist in the codebase]
    - Prototype: 7489862
8. Ifdef the platform-specific parts of `GlobalBrowserCollectionPlatformDelegate` in the header
    - Functionally equivalent to option 1, but the headers are mixed up with ifdefs in one file rather than separated into per-platform files. This fits with the `browser_window_interface.h` style, but I argue that it's harder to read (but perhaps more obvious what's going on?)
    - Precedent: shell_platform_delegate.h, renderer_main_platform_delegate.h, web_test_shell_platform_delegate.h,
    - Prototype: 7486835
9. Have subclasses of the `[Global|Profile]BrowserCollection` instead of using delegates at all

    - Intuitively “feels” like the wrong direction in a “prefer composition over inheritance” way
    - Seems common in the codebase, though
    - Needs static casting in `BrowserManagerService`, like option 4
    - Will need ifdefs or similar in `global_features.cc` to use the right platform impl
    - Precedent: tab_storage_packager_android.h, cert_verify_proc_android.h, tab_group_sync_delegate_android.h
    - Prototype: 7493267

- Haven’t looked *exhaustively*, but from what I’ve seen, the most common pattern in existing code is to not use a “platform delegate” and instead use inheritance to have per-platform subclasses (eg. at the `GlobalBrowserCollection` level - basically option 9), combined with inline ifdefs, and sometimes separate factory classes as well (having a factory seems like overkill to me, but maybe we’ll find that it helps for testing?). Examples: tab_storage_packager_android.h, cert_verify_proc_android.h, tab_group_sync_delegate_android.h
- Among cases where there is a platform delegate, mostly the pattern is that the header is shared between platforms (often with inline ifdefs) and separate .cc files (ie, option 8). I still maintain that option 8 is simply a less-readable version of option 1 or 2 in our case.

Decision: we will go with **option 8**