Chrome Sync Datatype API

Getting Chrome Services To Think Sync

Nicolas Zea
Fred Akalin

Overview

Google

- Sync Overview (The old way)
- SyncAPI++
- How to Transition
- Conclusion

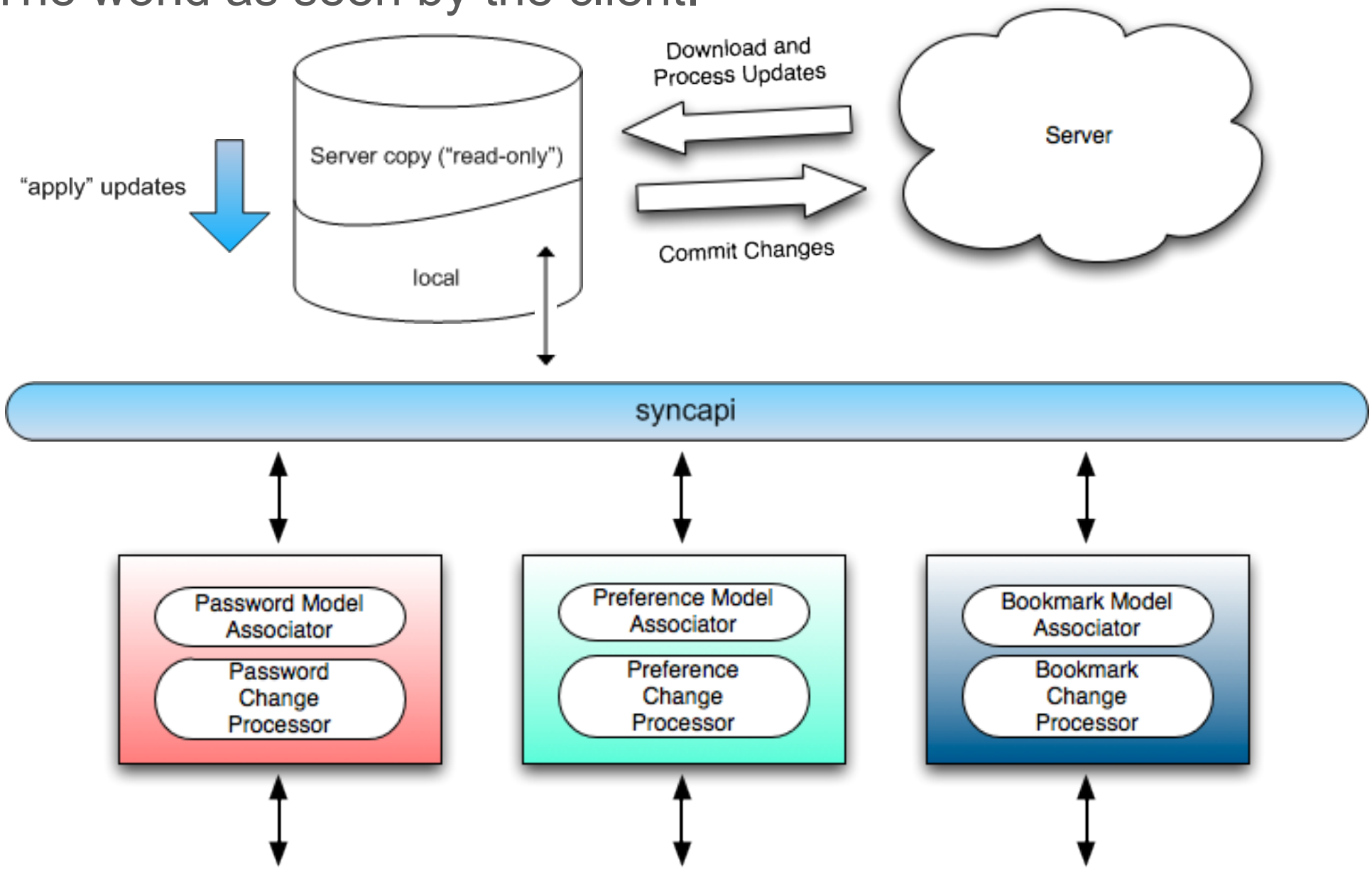
Sync Overview

Let's sync up about Sync.

Sync Overview

Google

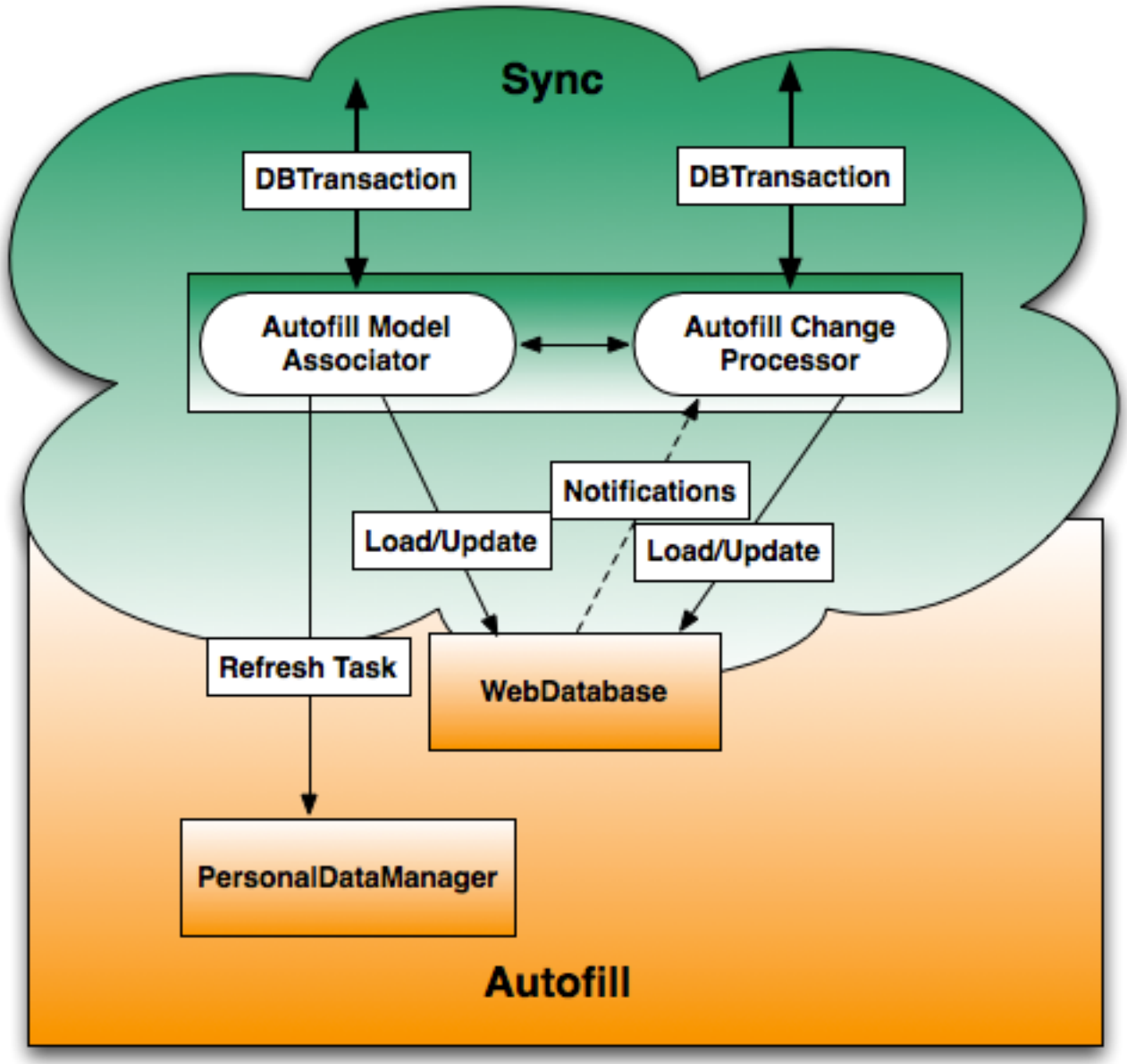
The world as seen by the client.



Sync Overview

Google

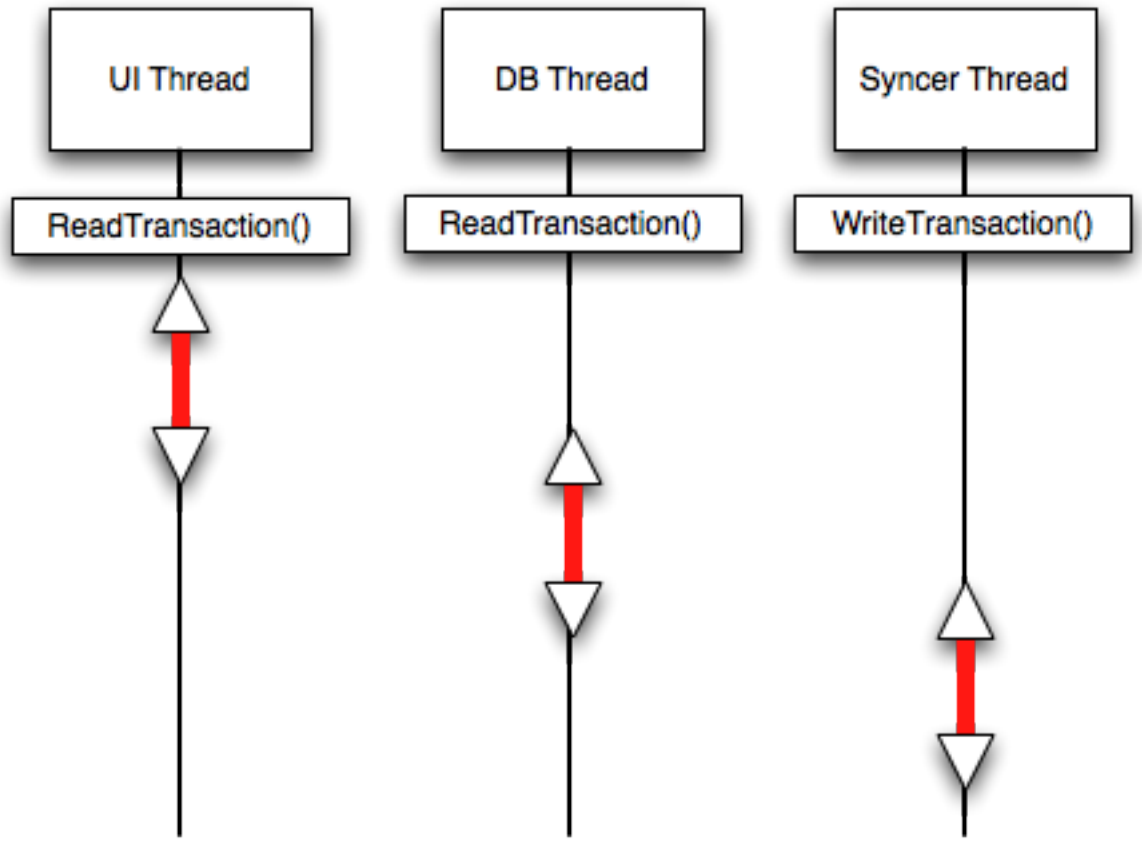
Complex interaction with chrome services.



Sync Overview

Google

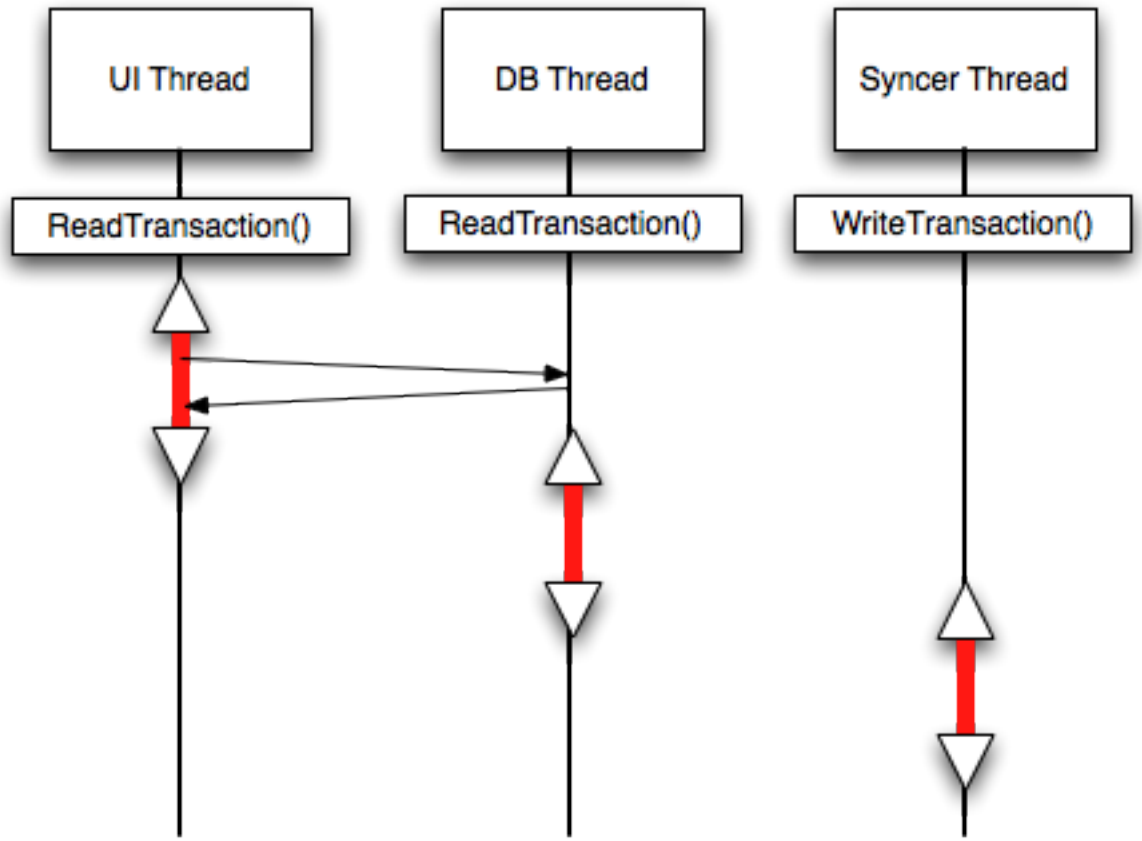
Transactional DB access.



Sync Overview

Google

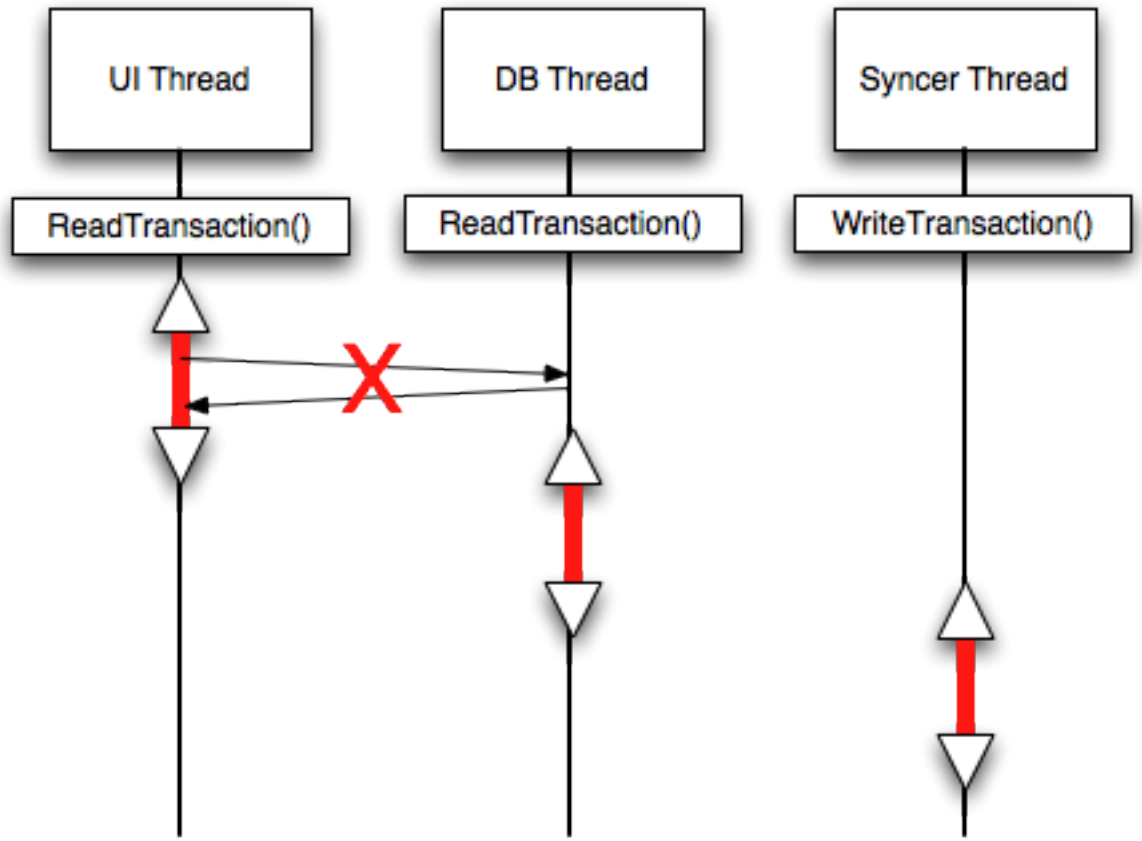
Transactional DB access.



Sync Overview

Google

Transactional DB access...and dangers.



Sync Overview

Google

Lack of clear ownership.

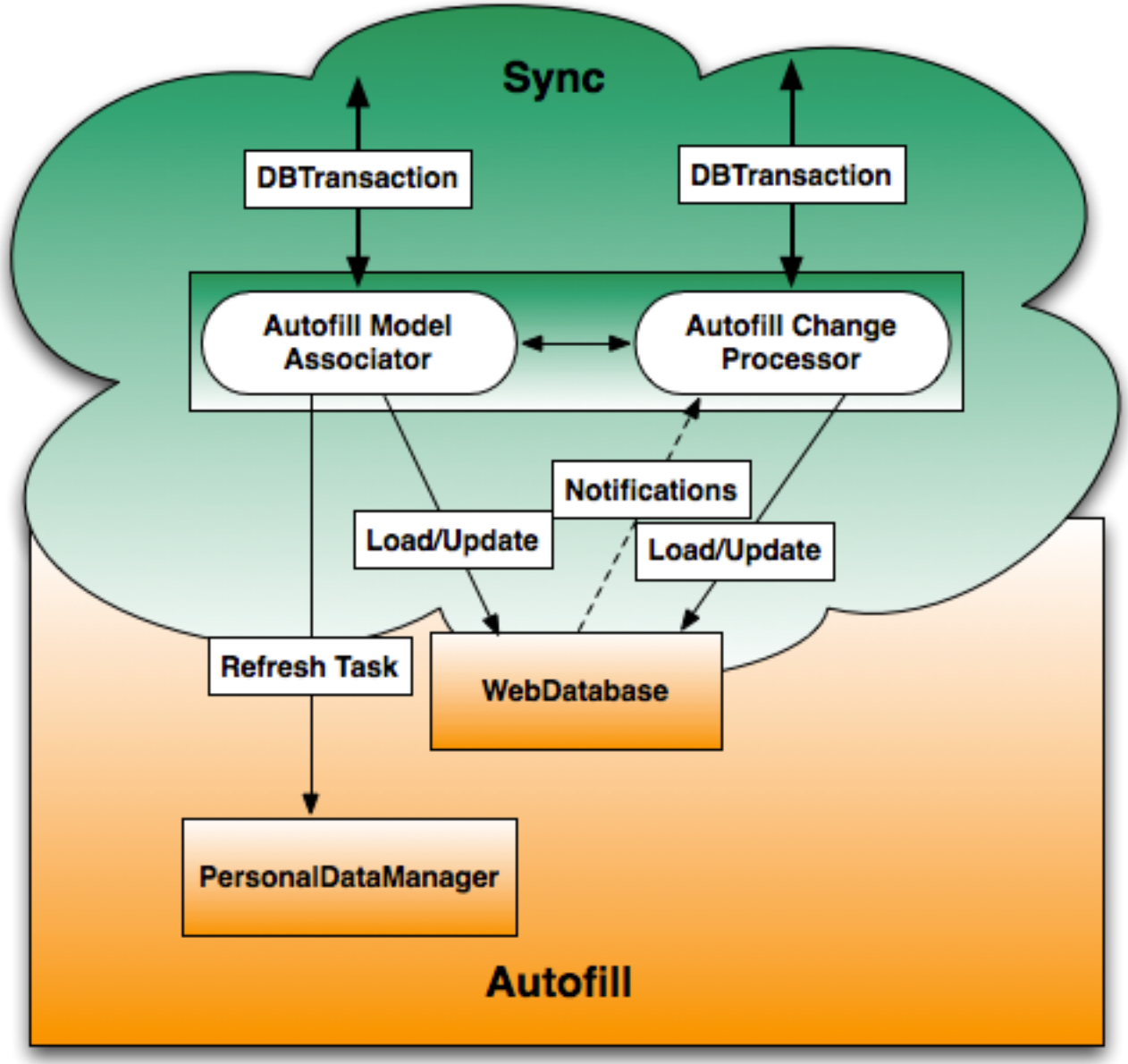
1. Sync owns sync functionality for a datatype
2. Datatype developers own chrome service functionality
3. Both developed in parallel and isolation
4. What could go wrong?

SyncAPI++

The New Way

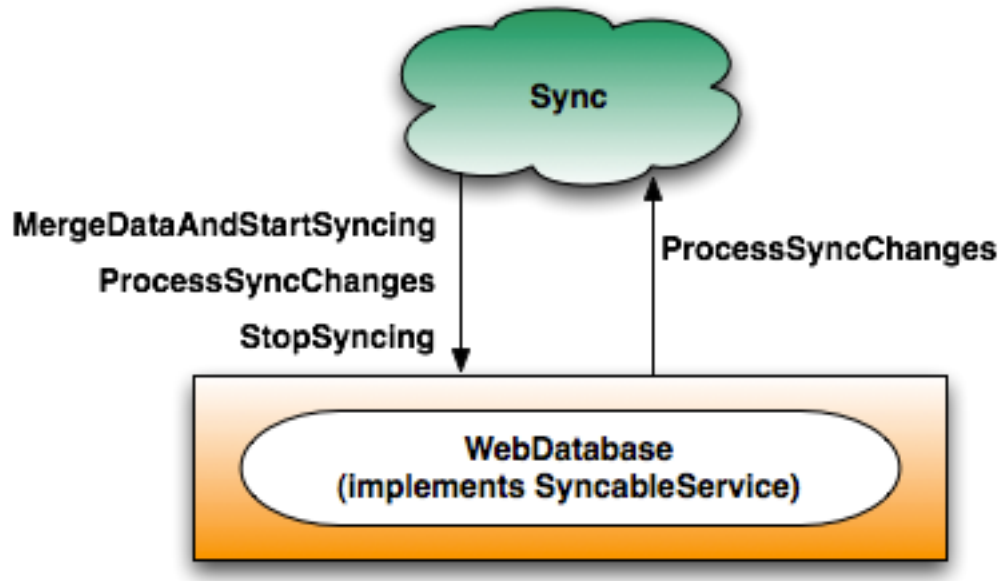
Sync Overview

How things were:



SyncAPI++

How we want them:



Chrome Services should speak sync.

SyncAPI++

Google

The new way:

- Simplified API
  - No worrying about transactions within datatype specific code
    - Transaction work is behind the scenes
  - Single SyncableService point of contact for every datatype
    - All datatype specific code related to syncing in one place
    - Thread-safe (should be on same thread as data)
- Chrome Service owns sync functionality
  - Can implement SyncableService directly
  - Datatype syncing integral part of service functionality

Simplified API

Google

SyncableService

- MergeDataAndStartSyncing
- StopSyncing
- GetAllSyncData
- Implements SyncChangeProcessor
  - ProcessSyncChanges

SyncChangeProcessor

- Interface for all sync changes to/from syncer
  - *SyncableService* invokes ProcessSyncChanges on it's sync_processor
  - Syncer invokes ProcessSyncChanges on each *SyncableService*

Simplified API

Google

SyncChange

- change_type - UPDATE,ADD,DELETE,INVALID
- *SyncData*

SyncData

- Opaque internal representation
- Contain all sync data
  - Protobuf representation
  - Unique client tag for each sync object
  - Origin - whether local service or syncer
- Immutable and lightweight copying
  - Can be passed by value and used in STL containers
  - Thread Safe

Simplified API

Google

```
bool MySyncableService::MergeDataAndStartSyncing(
syncable::ModelType type,
SyncDataList initial_sync_data,
SyncChangeProcessor sync_processor) {
                sync_processor_ = sync_processor;
        // Go through sync data and merge with local data.
SyncChangeList changes_to_sync;
for(...) {
SyncData sync_data = initial_sync_data[i];
DCHECK_EQ(syncable::MY_TYPE, sync_data.
GetDataType());
MyType converted_ata;
                // Convert from protobuffer to internal representation.
ConvertSyncSpecificsToMyType(initial_sync.GetSpecifics(),
            &converted_data);
```

Syncing a New Datatype

Google

10 Steps:

1. Get in touch with Sync Team :-)
2. Decide what data will be synced and how to represent it.
3. Protobuf specifications (See chrome/browser/sync/protocol/)
4. Datatype enum/helper routines (chrome/browser/sync/syncable/model_type.*)
5. Implement *SyncableService*
6. Test and iterate isolated SyncableService.
7. Datatype controller
8. Test and iterate SyncableService interacting with sync.
9. Test and iterate datatype interacting with server.
10. Profit.

Transitioning

How to painlessly (ish) implement new SyncAPI for old datatypes

Transitioning

Google

Main steps involved:

1. Merge model associator and change processor into a SyncableService
2. Remove sync_api::[Read/Write]Transaction usage, and replace with SyncData's and SyncChange's
3. Update ProfileSyncFactory to create SyncableServiceAdapter create appropriate GenericChangeProcessor

Transitioning

Google

Merging model associator/change processor into a SyncableService

- AssociateModels becomes MergeDataAndStartSyncing
- DisassociateModels becomes StopSyncing
- ApplyChangesFromSyncModel becomes ProcessSyncChanges
- Notification observer...killed
  - Service that stores the data should be Syncable
  - A ProcessLocalChange should be invoked directly

Transitioning

Google

Switching from sync_api transactions to SyncChanges

- No querying of the Sync DB anymore.
  - Receive initial sync state at startup time
  - Receive remote changes as they happen
  - Send local changes as they happen to syncer

- SyncableService should be able to reconstruct the sync data for that datatype from it's own local data
  - a) having all sync changes merged into it's local store
  - b) buffer any sync changes that aren't used
  - *GetAllSyncData*(...) should match what the syncer knows about

Conclusion

Google

- New SyncAPI greatly simplifies interaction with Syncer
- Chrome services should inherently "Think Sync" and interact with sync directly
- Control of datatype-specific sync logic is now responsibility of datatype
- We need your help moving old datatypes over to new API :-)

Questions?

Ready to sync?