Sync Policy

1. When an object is created or updated by the Client it saves the object locally in its SQLite database. The Client’s system timestamp is stored in the object’s “update” field and the object’s “dirty” field is set to true. If the network is available, the Client uploads the object to the Backendless server.
2. When Backendless receives an object and before it is “saved”, cloud code on the server searches its database using the object’s uuid.
   1. If the server object is not found (a newly created object), the object is **SAVED** TO BACKENDLESS DATABASE.
   2. If the server object is found, the cloud code compares the object’s server timestamp to the objects Client timestamp.
      1. If the Client’s timestamp is after the server’s timestamp, then the object is **SAVED** TO BACKENDLESS DATABASE.
      2. If the Client’s timestamp is before the server’s timestamp, then the object is not saved and Backendless returns an error message and the newer Backendless object to the Client. The Client saves the newer Backendless object to its SQLite database, marks its “dirty” field to false, and updates its UI.
3. SAVED TO BACKENDLESS DATABASE. Backendless saves the object in its database then returns the saved object to the Client. The Client saves the server’s timestamp, and if necessary, the object’s objectId to the Client’s SQLite database. The object’s “dirty” field is set to false. The Backendless server sends a “message” to any other client devices with the recently saved object.
4. When the Client’s “main activity” starts it syncs with Backendless by:
   1. Saving any Client dirty objects to Backendless, then
   2. Retrieving all Backendless objects from the server. The Backendless objects and Client objects are compared. Any new objects are inserted into the SQLite database. For objects that exist in both Backendless and the Client, the objects’ timestamps are compared. If Backendless objects timestamp is after local object timestamp then the object is updated in the Client’s SQLite database. Any objects that exist in the Client’s SQLite database but are absent from the Backendless objects are deleted from the Client’s SQLite database. The Client’s UI is updated.
5. When the Client’s “main activity” closes, it starts a service that saves all Client dirty objects to Backendless. If no network is available, the service continues to run. When the network becomes available, the dirty objects are uploaded to Backendless.