Offline Sync Checklist

What follows is a checklist for implementing sync in mobile devices with OutSystems. If you follow the checklist you will greatly improve synchronization of data between the mobile device and the server, as well as ensure there are no performance issues in the overall experience.

Performance

Follow these items to validate that your mobile app does not suffer from performance and usability issues that degrade the end-user experience.

Are the local entities designed as a lightweight version of the server entities? Have the entity relationships been denormalized?

Mobile devices do not have the network resources or computing power of a server, and can efficiently handle only a small amount of data. Make sure that the local entities contain only the attributes that your offline business logic requires. Do not add all the attributes of the server entities. Additionally, update only those records that are relevant to the current user of the mobile application.

Does the server logic minimize the data-set returned while keeping the logic simple?

The application should transfer the least amount of data that is needed to successfully complete the synchronization. Also, know your users and the conditions in which the app works - this will help you determine the optimal settings, such as the timeouts.

Is TriggerOfflineDataSync called to initiate the sync?

It is essential that all the elements that are used to manually start the sync call TriggerOfflineDataSync, which launches the sync in the background. TriggerOfflineDataSync is optimized for asynchronous sync, background tasks and triggering the sync events. Do not call OfflineDataSync to start the sync unless you want to block the flow execution and you do not need to trigger the sync events. Also, do not call Sync<LocalEntity> actions you may have defined.

Is there no logic that slows down the sync process?

Blocks such as Common/OfflineDataSyncEvents should not be edited. Do not add unnecessary calls to external services, untreated exceptions or server calls. This can slow down or sometimes completely freeze the app.

If you need to react to a network status change, do you use the NetworkStatusChanged block?

Don’t use heavy and complex JavaScript code to detect network changes and online status. Use NetworkStatusChanged block found in MobilePatterns/Private. There are also functions GetNetworkStatus() and GetNetworkType() to query the network in the action flows.
Do you use SyncUnit of the OfflineDataSync to sync only the entities that need an update?
The string parameter SyncUnit should be passed to the OfflineDataSync so the execution flow can branch and only sync the necessary entities.

Application Design

These items will assist you in creating a maintainable mobile application and deliver improvements over time.

Did you consider an app design that avoids conflicts during sync?
Sometimes the app can be improved so that conflicts between the local and server data never happen. Consider whether conflicts are essential to the use case. For example, maybe the last-write-wins is a pattern that suits most of the needs or editing can be locked while the device is offline.

If the business logic needs to deal with data conflicts, does the UX support the conflict management?
If data conflicts are part of your business case you should ensure that the user data is never lost because of a poor UX. The end-users should be clear about the consequences of the actions they choose and be informed of the sync status.

Do all synced local entities have their own action in OfflineDataSync folder?
Instead of placing all your logic for synced local entities in the default OfflineDataSync action, you should create individual actions in the same folder to sync each local entity. This helps to keep the sync logic organized and enables running syncs in a granular, controlled way.

Does the app clear the pending sync requests when needed?
The syncs started by TriggerOfflineDataSync wait in a queue until they run. There are some instances where a sync must run with a higher priority or you no longer need the currently queued syncs to run. To do this, call TriggerOfflineDataSync with the input DiscardPendingSyncUnits set to True to clear the sync queue.

End-User Experience

Follow these items to validate that you provide a solid end-user experience, in line with best practices.

Is automatic sync configured?
You may want to set up automatic data syncs for specific conditions in the client action OfflineDataSyncConfiguration, such as when the user logs in or when the app comes back online.

Does the app provide feedback to the end-users about the sync status?
The end-users should know about the status of the data sync. Use the Screen events to react to changes in the network and sync status and update the UI to provide visual feedback about what is happening. For example, display messages like "Syncing...", "Sync failed...", "You’re offline". You can also use the LayoutBlank block to display an error message and an option to retry.