

## Sync-Enabled Devices

# Table of Contents

<b>Overview</b> .....	3
<b>Syncable Device Applications and the Device Sync Agent</b> .....	4
· Basic Client Application Requirements .....	4
· Change Tracking – Dirty Bits vs. Change Counters .....	4
· Update/Delete Flag .....	5
· Field Level Change Tracking .....	5
· LUID's .....	6
· Soft Deletes and Filtering .....	6
· Private Records .....	6
· Record Versioning .....	6
· Sync Agent Requirements .....	6
· Sync Agent .....	7
· Sync Anchor .....	7
· The Sync Operation .....	7
· Sync Status .....	8
· Full Sync .....	8
· Time Zone .....	8
· Character Encoding .....	8
· Endian .....	9
· Purging .....	9
· System Capabilities .....	9
· API for Reading and Writing Records .....	9

## Overview

Waverley has now had significant experience developing sync agents for a variety of different devices. During this process, we have had the opportunity to examine how different devices build their operating systems and applications and how the choices made by the device designer effect the sync experience for the user. A few devices clearly have not considered sync before and others handle sync in a limited way. We often find ourselves working around limitations of a particular approach resulting in lost development time and user frustration. We have created this document to address the minimum required set of features needed to provide an acceptable sync experience and additional features necessary to create a world-class sync client.

# Syncable Device Applications and the Device Sync Agent

Providing a great sync experience for the user requires consideration from two distinct parts of the device. One is the support that the device applications provide for sync and the other is the support from the device specific Sync Agent.

Local applications store their state in a local device database. As the user makes changes and adds new data, database records are updated, added and deleted by the application. The syncing process requires that device databases synchronize with the current state of the sync server. After the sync is complete, the set of managed records are equivalent. In order to make the sync experience acceptable, the sync client needs a mechanism to identify records that have been changed and only transfer the changed data. Exchanging the entire database is inefficient and error prone and should be avoided where possible.

## Basic Client Application Requirements

The two most important requirements for enabling the sync are Change Tracking and Locally Unique Record Identifiers (LUIDS). Change Tracking requires a mechanism to identify which records have been changed by the user since the last sync and a flag to specify what kind of change was made. These are minimum requirements. To provide an enhanced sync experience, clients may choose to support a mechanism to handle Soft Deletes, Private records and record versioning.

## Change Tracking – Dirty Bits vs. Change Counters

There are several different methods for tracking changes in the device database. Logically, the device database has one or more columns containing data specific to the application. To enable sync, two additional separate logical columns are needed. One is required for recording the change version and the other is required to specify what kind of change is made. Only Updates and Deletes need to be recorded. Adds are treated as Updates by the Sync Client and the Sync Server resolves the Updates as Adds or Updates in the system. To record changes, some existing devices set a single bit when any field in the record has been changed. This bit is then cleared when the device is synced to a sync server. The single dirty bit mechanism can only support syncing to a single sync server. This is unacceptable because the Sync Agent can't support more than one sync server.

Once the sync is complete, the dirty bit must be cleared to prevent the same records from being synced again to the same Sync Server. However, some subset of the changed records may need to be synced to another Sync Server, which has a different state from the first Sync Server. For example, a device may need to sync wirelessly to at one time, and subsequently sync locally to a disconnected Desktop machine. To accomplish multi point sync, the Change Tracking column should contain a multi-bit numeric counter. The width of the counter depends on how many syncs are expected over the life of the database and how the Sync Agent handles rollover of the value. Generally, rollover will result in a full sync (where all the records are exchanged) because the state of the Change Tracker has been lost. We recommend at least two bytes for the Change Tracker value. As the client application makes changes to its local database, it marks the changed records with the Change Counter and thereby differentiates the modified records from ones exchanged during other sync sessions. During a sync, the Sync Agent can select the records that have been modified with the appropriate Change Counter to sync to the destination server. Note that the Sync Client does not need to (and must not) change the value stored in the Change Tracking column once the sync is complete. It simply updates the value that will be used to mark the next collection of changed records and new modifications are recorded in the database with the new Change Counter.

## **Update/Delete Flag**

The second required column indicates the type of change made to the database. The only differentiation required is Update vs. Delete. Note that records deleted by the client application must be recorded somehow so the Sync Agent can notify the Sync Server of the delete and generate an equivalent record set. The records cannot simply be deleted and forgotten. The Sync Agent is responsible for Purging these records once successful notification of the delete has been sent to the Sync Server. The client applications have the choice of using the Delete flag to identify deleted records or may choose to delete these records from the main database and record the deleted state (and potentially the data itself) in a separate database. Most devices use a flag to record deleted records and do not move them to a different database. However, this requires that the managing application be aware of the deleted records and not display them to the user, and omit them from records counts, sorted lists, etc.

## **Field Level Change Tracking**

Change Tracking on the record level is the minimum requirement for a syncable application. However, field level changes are desirable. The tradeoff is the space required to store the

extra change information on the device vs. the time and space required to exchange changed information with the Sync Server. If it is possible to send only field level changes, it reduces the size of the sync packets and simplifies the task of the Sync Server when it attempts to resolve differences between the state of the device and the Sync Server.

## LUID's

In addition to a Change Tracking mechanism, each record in the Device Database needs a locally unique record identifier or LUID. The LUID is local to the device and must uniquely identify a record. LUIDs are necessary because the sync architecture cannot rely on matching records solely by name or other data specific to the record. LUIDs are exchanged between the device and server and the server keeps a mapping table between the Sync Server's Global Ids and the LUID. When a new record is added by the server, the device performs the add and a LUID is assigned to the new record by the client application managing the device database. The new LUIDs is then mapped back to the server so that the device record and Sync Server record match can be made again in the future.

## Soft Deletes and Filtering

Devices may choose to either soft delete or hard delete records. Soft deletes are only removed from the device, not from the Sync Server. Hard deletes are removed from both the device and from the Sync Server.

## Private Records

The user may wish to maintain records that are excluded from the sync process. These are called Private Records. If the device supports Private Records, the Sync Agent must be aware of them and exclude them from the sync.

## Record Versioning

Record versioning may be necessary depending on what the expectations are for data exchange between the Sync Agent and the Sync Server.

## Sync Agent Requirements

The Sync Agent is an application that performs the sync. It works in conjunction with the local device applications during the sync process. The Sync Agent needs to provide a

Sync Anchor, Sync Status flags for the records to sync and a record reading API in order to complete a synchronization session. The agent needs to support Full Sync on an app specific basis. The agent needs to be aware of time zones, character encoding, endian issues to work properly. Lastly, the agent needs to know when to purge deleted records, how to report errors back to the server and how to query the system for capabilities, especially if the device supports filtering.

## Sync Agent

The user may wish to maintain records that are excluded from the sync process. These are called Private Records. If the device supports Private Records, the Sync Agent must be aware of them and exclude them from the sync.

## Sync Anchor

The Sync Anchor works in conjunction with the Change Tracking mechanism described earlier. The Sync Anchor is a number maintained by the Sync Agent and is separate for each Sync Server. Furthermore, the Sync Anchor must also be separate for each application (dataclass) and for each Sync Server, if the Sync Agent allows the user to control which applications are synced. If no such user control is provided, then the Sync Agent can assume that all syncable application databases are synced during each sync session and therefore a database specific Sync Anchor is not necessary.

## The Sync Operation

The Sync Agent performs the sync process by selecting and exchanging the set of records from the application databases whose Change Tracker value is equal to or greater than the Sync Anchor for the destination Sync Server. The Sync Agent must also handle adds or updates coming to the device from the Sync Server. Adds and updates coming from the Sync Server are marked with a Change Tracker value which is equal to the largest Sync Anchor. This prevents the changes coming the Sync Server from being synced back to the same Sync Server, but they will be synced to a different Sync Server during a future sync session. Once the sync is finished, the Sync Client updates the destination Sync Anchor with a value one greater than the largest value in any of the Sync Anchors. The Change Tracker values recorded for each record in each application database are not changed by the Sync Agent.

## Sync Status

During the sync process, records are read from the application databases and sent to the Sync Server. It is possible that an error can occur or that the connection is lost and the sync needs to be restarted. In this case, the Sync Agent does not know which records were already sent in a previous attempt and which records have not yet been sent. To solve this problem, the Sync Agent sets a sync status flag for each record to be synced to the server. As records are successfully sent, the Sync Agent clears the flag. If the sync needs to be restarted for some reason, the Sync Agent can check to see if any syncable records have a status flag turned on and sync those records. When the Sync Agent gets ready to start a sync, it must first determine if a partial sync was previously in progress. If not, the regular process of setting all the Sync Status flags happens first and the records are sent sequentially. The Sync Status flag is used by the Sync Agent and can be a separate entity (eg, an array) or can be a column in the local application database.

## Full Sync

Full Sync is the process of exchanging all the data on the device with the Sync Server. It is a process that should be avoided because it is time consuming and expensive if over a wireless connection. However, full sync is required under certain situations: device for the first time, device is hard reset, or some other situation where an unrecoverable error occurred and the only way to ensure equivalent records sets is to full Sync the client device and the Sync Server. The Sync Agent should make a full sync option available to the user.

## Time Zone

Time zone is an issue and important issue for Sync Clients. Potential problems arise when syncing a local time device to a server that recognizes UTC times. This can result in duplicate records for data containing time information.

## Character Encoding

The Sync Server will compare data being sent from the Sync Client during the sync process. Character encoding is an important issue for the comparison.

## Endian

The Sync Server must use the same byte order for comparing numeric values coming from the Sync Client.

## Purging

The Sync Agent is responsible for purging local application database entries marked as deleted by the managing application. Once the sync is complete, the Sync Agent can delete records whose Change Tracker value is less than the minimum of all the Sync Anchors.

## System Capabilities

The Sync Agent may need to query the device to find out about its capabilities. For example, many devices support a limited amount of data. The Sync Client needs to relay this information to the Sync Server to meet the expected behavior during a sync.

## API for Reading and Writing Records

The Sync Agent needs to be able to read and write the local application databases during sync. It is desirable to have an API to perform the reads and writes to encapsulate the access functionality and to ensure integrity and relations between data.