# How to implement product subscription bussiness model with Nokia In-application Purchase API

This article explains how to ... **implement subscription model in application by extending example application project existing in IAP SDK for Qt**

## Introduction

Book store example application can be found in In-app Purchase for Qt by the path after installation \iap_qt_sdk_0.3.1_24.01.12\Examples\InApplicationPurchase\BuyAndDownloadExample

This application project demonstrates how to implement In-application Purchase with your own access control to protected resources that are stored on third party server. Third party server can validate user requests for a product with OVI Store via Purchase Ticket Verification API. Book store example application functionality allows user to buy a product and use it forever. **How to with minimal efforts implement subscription model on top of Book store example application?**

## Installation package and source code

Please find the application package and source code linked to this article properties. **Note: the application works in IAP test mode -- so no real money transaction happens**. To easy explore the project source code it would be the best approach to compare code delta between original project BuyAndDownloadExample and this one. Project comparison is especially helpful for mobile application client code (directory name is 'BuyAndDownloadExample_10_minutes_suscription\code\application') because the original code is not quite clear to understand and new functionality on top of it makes it even harder to understand due to functionality reuse. Try to focus on 'Implementation approach' chapter to grasp the idea.

## Implementation approach

### architecture

When user requests server for a product, server checks whether this product has already been bought by this user. To do this checking server implementation **dbaseManager** has function

```
public function getTicket($var_account, $var_prodid)
```

that returns purchase ticket or empty string.

Before the ticket existence check we are going to introduce **Subscription expiration check**. Upon the Subscription expiration checking result user's purchase ticked may be deleted if it is found to be expired. Then if the ticked record has been deleted in Subscription expiration check, **dbaseManager.getTicket** will return empty string indicating user has to buy that product.

The rest of Book store example application business logic will remain unchanged.

### performance consideration

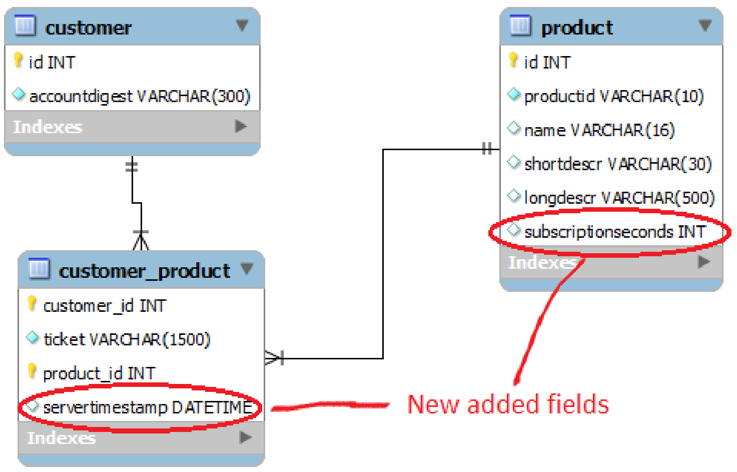
For performance reason subscription expiration check for a product will be performed only for user who requests this product -- i.e. we don't scan the whole purchase list for purchase expiration.

## Implementation

1. We need to introduce expiration timer to the product catalog database implementation. Stored procedure will delete record if ticket timestamp is older that it is specified in 'subscriptionseconds' filed. Seconds unit is taken to operate with unix time stamp

```
CREATE PROCEDURE deleteExpiredSubsription (IN var_account VARCHAR(300), var_prodid
VARCHAR(10))
BEGIN
 DELETE FROM customer_product WHERE ticket IN ( SELECT * FROM (
  SELECT t.ticket
```

```
   FROM customer c
   INNER JOIN customer_product t ON c.id = t.customer_id INNER JOIN product p ON
p.id = t.product_id
   WHERE p.subscriptionseconds IS NOT NULL AND
    UNIX_TIMESTAMP() - t.servertimestamp - p.subscriptionseconds > 0  AND
     c.accountdigest = var_account AND
      p.productid = var_prodid
  ) AS tempTable );
 -- about 'tempTable' above please check http://www.mysqlfaqs.net/mysql-
faqs/Errors/1093-You-can-not-specify-target-table-comments-for-update-in-FROM-
clause
 -- the point is : MySQL does not allow to UPDATE or DELETE a table's data if
you're simultaneously reading that same data with a subquery.
 -- Error #1093 workaround
END$$
```



2. We need to check whether subscription is expired in the mobile application code. When user tries to open already bought product subscription checking request will be issued to server. According to the implementation approach, when the server receives product request, first it checks user's ticket for expiration and then returns either ticket value or empty string. If empty string is returned, mobile application logic initiates purchase flow.

```
void MainWindow::playLevel()
{
    .........
    // before opeining downloaded product check with server back-end whether its
ticket has expired
    catalog->checkSubscriptionExpiration(finfo.baseName()); //product saved in
file system takes product ID filename
    .........
}
```

Method 'catalog->checkSubscriptionExpiration' breaks previously implemented use case when user could open the product and browse its content. Now checkSubscriptionExpiration requests the server for product url passing user authentication data and product ID. As it is mentioned above, server receives product request and makes checking for subscription expiration first if the product ticked found in the database. It can delete this purchase record in the database if it is found purchase has already expired, then original project functionality works.

## Summary

Nokia In-application Purchase API non-DRM product protection model allows developers to implement highly customized business logic in application. In this example application project shows how to add functionality of product subscription expiration in already existed project.