In-App Purchase Programming Guide

A method using AppControl for Tizen

Table of Contents

Abstract

Introduction

Selling items from within your application can provide large revenue. According to information and analysis provider IHS Screen Digest It will rise to more than $5.6 billion in 2015. You can implement In-Application Purchases in your application using the AppControl mechanism for Tizen Native and Web applications.

For example, you could use IAP in any of the following scenarios:

- An application that enables additional features
- Any application that allows the user to remove adverts
- An audio-book application that allows the user to purchase and download new books
- A game which offers new levels to play
- A RPG game that allows to buy virtual items
- And many other features…

What is the In-App Purchase AppControl mechanism?

If you want to sell items inside your applications such as in-game coins, levels and others, you may need to set servers up to communicate with, you may need billing infra to gather payments, you may need to manage interactions with users for authentications, payment and many other things. In-App Purchase will simply handle all those things mentioned above and you don't need to worry about it.

What you need to do is to prepare your applications for communication with the IAP and communicate with it. For example, with in-app purchase of items, your applications just find the proper AppControl, starts the purchase method and receive the result of the purchase in the method of the listener method. IAP will show dialogs (in case of errors), communicate with Tizen Stores server and invoke the listener methods to return the result of the request.

This guide will explain the process of using the IAP from A to Z. Please read this document thoroughly before you use it.

Contents of the distributed package

IAP is distributed as a ZIP archive. Archive contains following files and directories:

- Sample_Native: A sample Tizen Native application project in form of Tizen IDE project with source code.
- Sample_Web: A sample TizenWeb application project in form of Tizen IDE project with source code.
- docs: Directory with all documentation related to IAP including this guide.

Terms and abbreviations

IAP	In-App Purchase
MCC	This code (3 digits) identifying Mobile Country Code.
MNC	This code (chars/digits) identifying carrier in current country.

Applying IAP to your application

Applying IAP is not that hard because it is very simple and lightweight. There are these steps to follow:

① Register your items to TizenStore Seller Office. http://seller.tizenstore.com)
② Program your application to work with IAP.
③ Test and upload your application.

1. Registering your items to TizenStore Seller Office

Before you set up and program your application for IAP, you need to register item group and items to TizenStore Seller Office. The item group is a collection of items and it will be connected to your application in TizenStore Seller Office. For example, if there are 10 kinds of special items you need for your game, you need to create one item group for your game and 10 individual items. Registering is really simple.

Connect to TizenStore Seller Office.

Run a web browser and connect to TizenStore Seller Office. The address of TizenStore Seller Office is http://seller.tizenstore.com . Log in to TizenStore Seller Office. If you don't have any account for TizenStore Seller Office, you can make your own. After you have logged in to the TizenStore, If you want to sell applications under your personal name, you should register as a private seller or, if you want to sell applications under your company name register as a corporate seller.

Register a New Item Group

Follow these steps to register a new item group on Seller Office.

① Click 'Applications > Item' to navigate to the selected menu option.
② Click [Add Item Group].
③ Fill in the fields for the Item Group Title and Description, and click [OK] to register your item group.
④ When the item group is registered, you can click [Edit] to modify its information.
⑤ Select a checkbox and click [Delete] to delete an item group. When you delete an item group, all the items included in that group will also be deleted.
⑥ You can copy registered item groups from the item group list. When you copy an item group, all the items included in that group will also be copied.
⑦ An Item group can be modified, deleted, or copied when the application has the following statuses.

Add a new item (temporary status)

From the item list, select the name of a registered item group to navigate to its information page. After an item group is created, a list of temporary tabs will appear on the information screen when you first access it.

① Click [Add Item] in the temporary tab on the view page to enter basic information, such as Title, Price, and Description, in the Add Item popup window. Then click [OK] to add items. See the [Application Registration Guide] for details on price setting.
② You can batch register items by clicking [Item Bulk Upload] to upload an Excel file.
③ When the item has been added, you can click its title in the item list to view the added content.

1) Edit an Item (temporary status)

You can add, modify, or delete an item if there is an application listed in the item group, or if registered or revised applications in the item group are listed under Pre-Certification.

① You can click the title of the item on the temporary tab list and check or edit the item title, price, description, and image in the View popup window.
② If all the applications containing the item are listed under Pre-Certification, select the checkbox and click [Delete] to delete the item from all of them. However, items mapped for applications that are for sale cannot be deleted.
③ All changes will be applied when all the mapped applications are validated. The [On Standby] button is displayed until validation is complete.
④ Item statuses for different types of application status are as follows:

2) Item sales (sales status)

① Items are registered and modified and you can view them in sales.
② If all the mapped applications that are being registered or revised in the item group are listed under Pre-Certification, click the [Modify Item(s)] button to create a temporary tab that will allow you to modify the applications.

2. Setting your application project up to use IAP

There is nothing additional to be done in the project to make your application work with IAP. It uses an AppControl mechanism, and as such requires no updates to the application project.

Working with IAP means simply getting the proper AppControl.

One can find the AppControl tutorials in Help section of TizenIDE. Or directly you can refer to the link:

- **Application Controls**
- **AppControl class reference**

3. Programming your application to work with IAP

Programming with IAP is very similar to programming with Native Tizen API and other AppControls supported in the system. Before you implement your code you should edit the manifest file first.

Note that the IAP allows for two modes of operation (key name “_mode”):

- normal – commercial mode, to be used with the finished application, released on the market. Proper data will be obtained and returned from the IAP server.
- developer mode, always success – can be used while development of the application for testing purpose. Payment will always succeed.

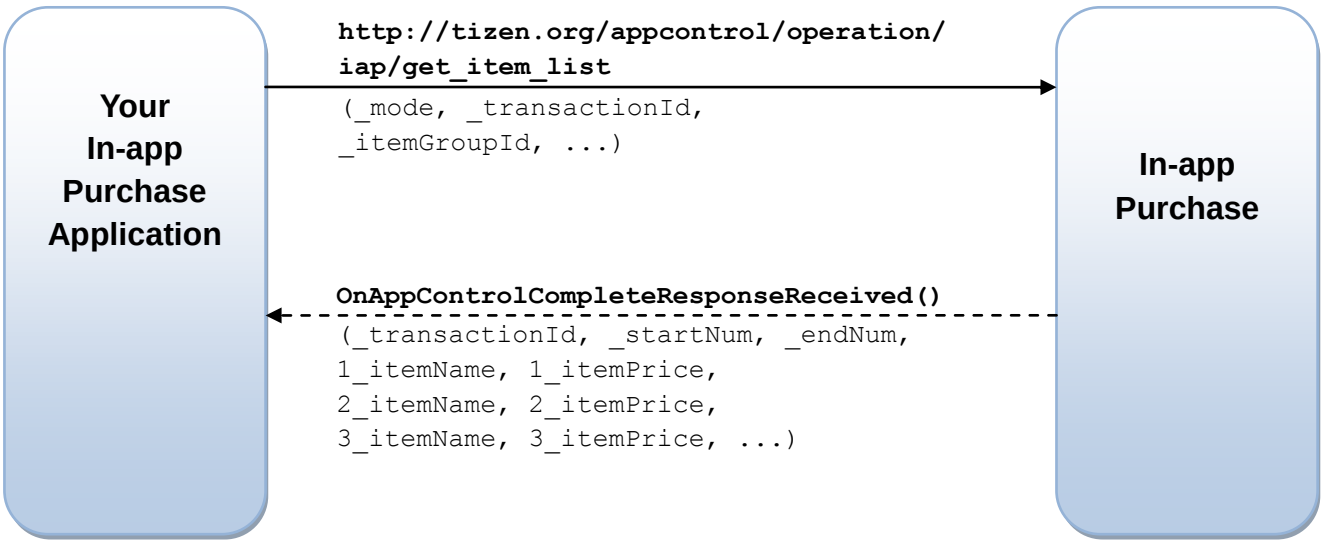
Following table describe the Interfaces list

Application ID	Operation ID	Description
`tizeninapp.IapClient` (Purchase initialization)	`http://tizen.org/appcontrol/operation/iap/purchase`	The operation makes a purchase of Item. It shows a purchase screen for an item. There are two steps: initialization and finalization. During this step a purchase screen will be displayed. And the user will be able to provide his details (e-mail, password) to make a purchase. The results of the operation are returned in the `OnAppControlCompleteResponseReceived()` event handler. The output data values are used to verify the payment and to proceed with purchase. During this step your application will receive a `_ticketVerifyUrl` to verify the purchase with Tizen Store IAP Server and `_ticketPurchaseId` to resume the purchase (continue with step 2).
`tizeninapp.IapClient` (Purchase finalization)	`http://tizen.org/appcontrol/operation/iap/purchase`	Second step is finalization. During this step actual payment will be made, and the user will be charged. The results of the operation are returned in the `OnAppControlCompleteResponseReceived()` event handler. The output data values are used to sent to your app information about status of purchase (if it is success or not).

`tizeninapp.IapService`	`http://tizen.org/appcontrol/operation/iap/get_item_list`	The operation returns a list of item available for purchase. The results of the operation are returned in the `OnAppControlCompleteResponseReceived()` event handler. The output data values are used to send a list of items available for purchase.
`tizeninapp.IapService`	`http://tizen.org/appcontrol/operation/iap/get_purchased_item_list`	The operation returns a list of already purchased items. The results of the operation are returned in the `OnAppControlCompleteResponseReceived()` event handler.
`Tizeninapp.IapService`	`http://tizen.org/appcontrol/operation/iap/get_country_list`	The operation returns a list of counties' mcc and mnc codes to be used in developer mode during testing in-app purchases. The results of the operation are returned in the `OnAppControlCompleteResponseReceived()` event handler.

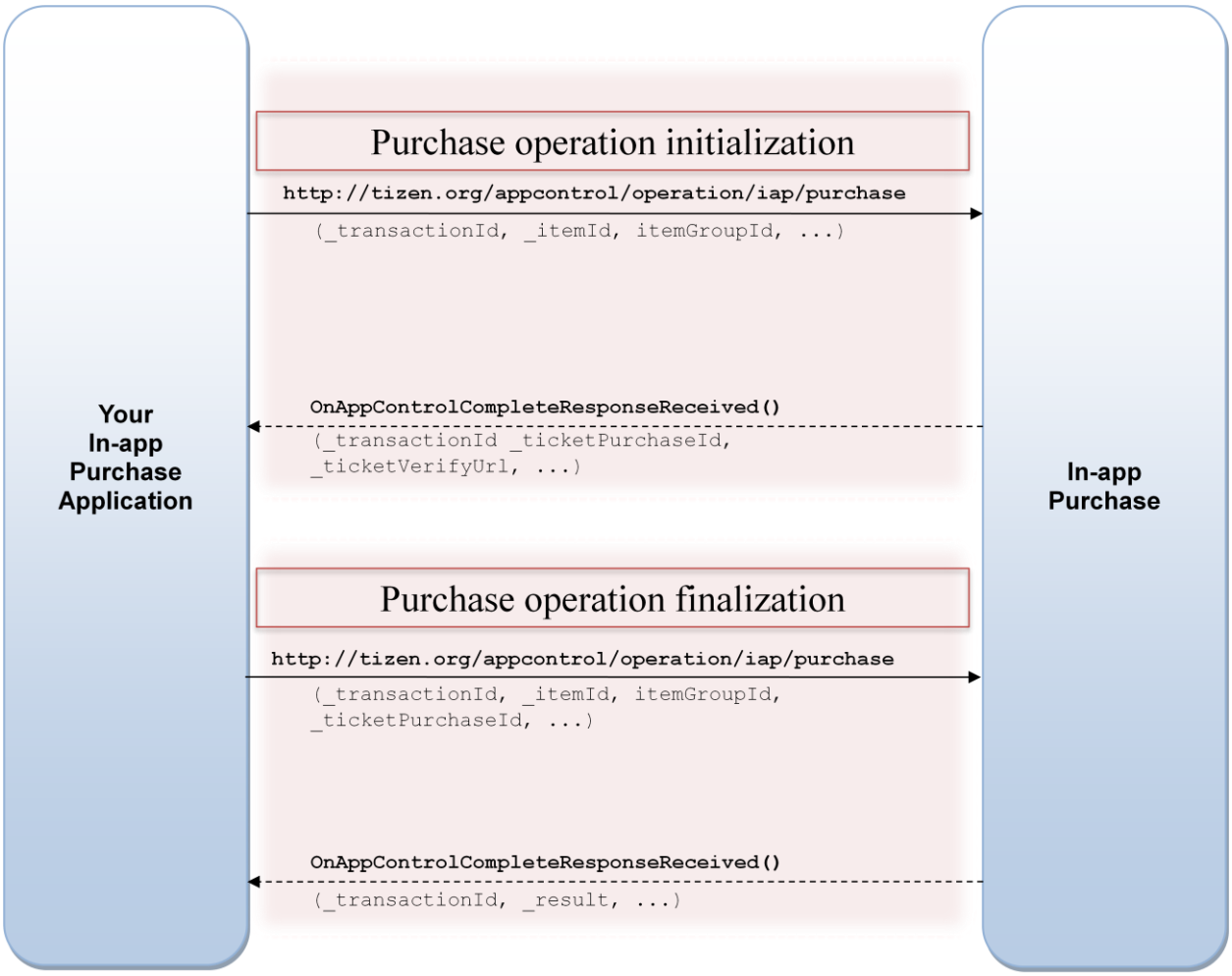
Purchase Item flow:

1. Get Item list : Use the `tizeninapp.IapService` ApplicationID to retrieve a list items from Tizen Store IAP Server for a given groupId:



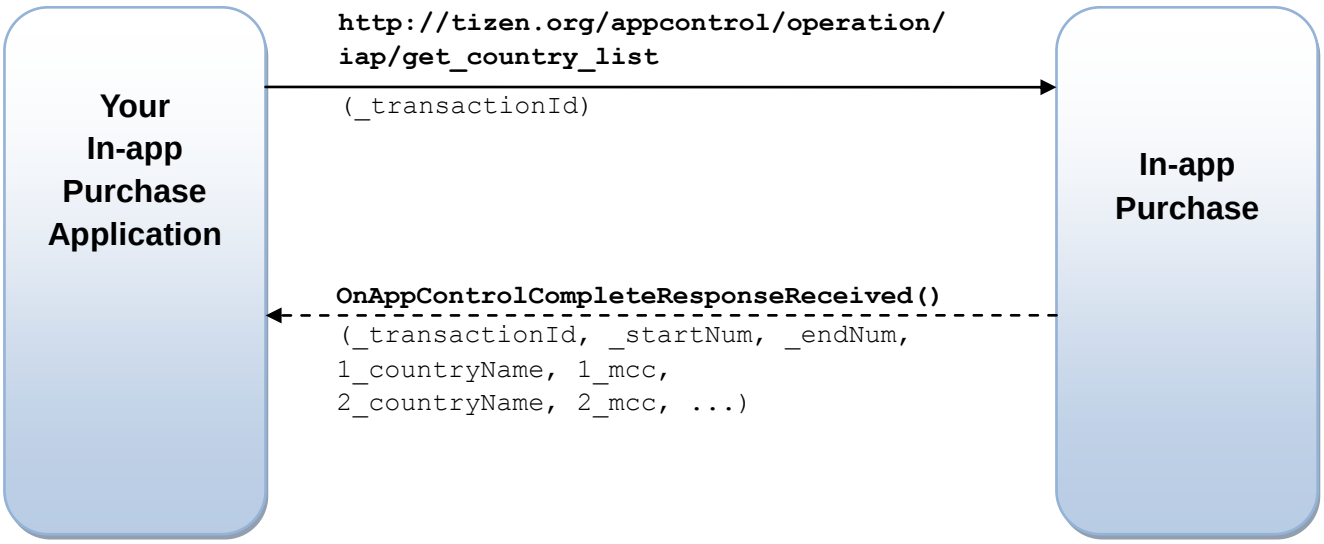
2. Display a list of items in Your In-App Purchase Application.

3. Purchase the item : Use the `tizeninapp.IapClient` Application ID to make a purchase of particular item. Purchase happens in two steps. Purchase initialization and purchase finalization.



Get a list of country

In developer mode you are able to test In-App Purchase with different country servers. To get a list of available servers you can use `http://tizen.org/appcontrol/operation/iap/get_country_list` operation:



3.1. Add permissions to manifest file.

IAP uses AppControl interface to handle purchases. To add required permissions to your application project, open manifest.xml file and add following permissions:

- http://tizen.org/privilege/application.launch

Below is the content of the manifest file which contains required permissions.

```
<?xmlversion="1.0"encoding="UTF-8"standalone="no"?>
<Manifestxmlns="http://schemas.tizen.org/2012/06/manifest">
    ...
<Apps>
        ...
<Privileges>
<Privilege>http://tizen.org/privilege/application.launch</Privilege>
</Privileges>
        ...
</Apps>
</Manifest>
```

3.2 IAP Service Control (Get item list, get purchased item list, get country list)

The IAP Service Tizen::App::AppControl instance allows you to get an list of items available for purchase and to get an list of already purchased item. It also allows you to get an list of countries available for testing in developer mode

Application ID

This application control can be accessed using aliased application ID of "`tizeninapp.IapService`".

Operation ID

This application supports the `http://tizen.org/appcotnrol/operation/iap/get_item_list` and `http://tizen.org/appcontrol/operation/iap/get_purchased_item_list` operations. It also supports `http://tizen.org/appcontrol/operation/iap/get_country_list` operation.

GetItemList operation

This operation returns a list of items available for purchase.

Input data

The following table show the (key, value) pairs required in the input extra data for the `http://tizen.org/appcotnrol/operation/iap/get_item_list` operation:

Key	Value	Description
`_mode`	`0 | 1`	Mode type. This information is optional. The default value is `0` `0` is normal (commercial) mode. This mode need to be used in application submitted to Tizen Store. `1` is developer mode (always success) This mode can be used while development of the application for testing purpose. Payment will always succeed.
`_transacionId`	Transaction ID number, such as `1, 2, …`	Transaction ID. This information is mandatory. This is used to track a transaction between requests.
`_startNumber`	Index of the first item	Index of first item on the list This information is mandatory Start downloading item from this numnber
`_endNumber`	Index of the last item	Index of last item on the list This information is mandatory. Stop downloading items up to this number. In the below response example, even for the case when the item information list has over 100 items, it only requests 15 items from the first one.

Key	Value	Description
_itemGroupId	ID of Group, such as 100000001455	Group ID Group ID is a ID associated with the particular Group of items in the Tizen Store Seller Office site. You need to register your Group Id on the Tizen Store Seller Office website first.
_mcc	MCC Code, such as 250	MCC Code This information is not mandatory. It can be used only in developer mode. You can receive a list of available MCC codes through GetCountryList operation
_mnc	MNC Code, such as 01	MNC Code This information is not mandatory. It can be used only in developer mode. MNC Code is Mobile Network Code.

Example code for GetItemList

```
HashMappArgList;
pArgList.Construct();

String key_mode = L"_mode";
String val_mode = L"0";

String key_transactionId = L"_transactionId";
String val_transactionId = L"1";

String key_startNumber = L"_startNumber";
String val_startNumber = L"1";

String key_endNumber = L"_endNumber";
String val_endNumber = L"15";

String key_itemGroupId = L"_itemGroupId";
String val_itemGroupId = L"100000001455";

pArgList.Add(&key_mode, &val_mode);
pArgList.Add(&key_transactionId, &val_transactionId);
pArgList.Add(&key_startNumber, &val_startNumber);
pArgList.Add(&key_endNumber, &val_endNumber);
pArgList.Add(&key_itemGroupId, &val_itemGroupId);

AppManager* pAppManager = AppManager::GetInstance();
AppControl* pAc = pAppManager->FindAppControlN(L"tizeninapp.IapService",
      L"http://tizen.org/appcotnrol/operation/iap/get_item_list");
```

```
pAc->Start(null, null, &pArgList, this);

delete pAc;
```

Output Data

The results of the operation are returned in the `OnAppControlCompleteResponseReceived()` event handler.

The following table shows the output data for the `http://tizen.org/appcotnrol/operation/iap/get_item_list` operation:

Key	Value	Description
`_method`	`OnItemInformationListReceived`	Method to be called as a purchase request result. This value will be `OnItemInformationListReceived`
`_result`	`<Result code>`	Result code number. Error codes are described in the end of this section.
`_transactionId`	`<TransactionID>`	Transaction ID number This ID will be the same as a Transaction ID used during request.
`_startNumber`	`<start number>`	Start Number Index of the first item on the list
`_endNumber`	`<end number>`	End Number Index of the last item on the list
`_totalCount`	`<total count>`	Total items count Number of items based on the startNumber and endNumber

We also have _totaCount number of items in a output data, each key is a PREFIX (list item index value) plus a key (i.e. “12_itemId”) as below:

Key	Value	Description

Key	Value	Description
PREFIX_itemId	<ItemID>	Item ID number This ID will be the same as a Item ID used during request.
PREFIX_itemGroupId	<ItemGroup ID>	Item group ID Item Group ID is a collection of items and it will be linked to your application in Tizen seller site.
PREFIX_itemName	<Item Name>	Item Name Item Name is a name provided during item registration on the Tizen Store Seller Office.
PREFIX_currencyUnit	Currency unit, such as: $,Won, Pound	Currency Unit Device user currency unit.
PREFIX_unitPrecedes	0 | 1	Unit Precedes String representation of values: 0 : Tall (2.99 TL) 1 : Front ($ 2.99)
PREFIX_hasPenny	0 | 1	Has Penny Informs if currency unit has penny: representation 0 : no 1 : yes
PREFIX_itemPrice	<Item price>	Item price This is a price of item in local currency.
PREFIX_itemDownloadUrl	<URL>	Item download URL This is a URL provided during item registration on the Tizen Store Seller Office.

Key	Value	Description
PREFIX_itemImageUrl	<URL>	Item image URL This is a URL provided during item registration on the Tizen Store Seller Office
PREFIX_itemDescription	<Item description>	Item description A description provided during item registration.
PREFIX_reserved1	<Reserved Field 1>	Reserved Field 1
PREFIX_reserved2	<Reserved Field 2>	Reserved Field 2

Example code for retrieving a list of items

```
void BuyEngine::OnAppControlCompleteResponseReceived(const AppId &appId, const String &operationId,
    AppCtrlResult appControlResult, const Collection::IMap *pResultMap)
{
if(pResultMap->GetValue(String(L"_method")) != null)
{
        String start = *((String*)(pResultMap->GetValue(String(L"_startNumber"))));
        String end = *((String*)(pResultMap->GetValue(String(L"_endNumber"))));
        intstartNum, endNum;
        Integer::Parse(start, startNum);
        Integer::Parse(start, endNum);
        for (int i=startNum; i<=endNum; ++i)
    {
        String prefix;
        prefix.Append(i);

        String key_itemId = prefix;
        key_itemId.Append(L"_itemId");
        String itemId = *((String*)(pResultMap->GetValue(key_itemId)));

        String key_itemName = prefix;
        key_itemName.Append(L"_itemName");
        String itemName = *((String*)(pResultMap->GetValue(key_itemName)));

        . . .
    }
}
}
```

GetPurchasedItemList operation

This operation returns a list of already purchased items.

Input data

The following table show the (key, value) pairs required in the input extra data for the `http://tizen.org/appcontrol/operation/iap/get_purchased_item_list` operation:

Key	Value	Description
`_mode`	`0 | 1`	Mode type. This information is optional. The default value is `0` `0` is normal (commercial) mode. This mode need to be used in application submitted to Tizen Store. `1` is developer mode (always success) This mode can be used while development of the application for testing purpose. Payment will always succeed.
`_transacionId`	Transaction ID number, such as `1, 2, …`	Transaction ID. This information is mandatory. This is used to track a transaction between requests.
`_startNumber`	Index of the first item	Index of first item on the list This information is mandatory Start downloading item from this numnber
`_endNumber`	Index of the last item	Index of last item on the list This information is mandatory. Stop downloading items up to this number. In the below response example, even for the case when the item information list has over 100 items, it only requests 15 items from the first one.
`_itemGroupId`	ID of Group, such as `100000002501`	Group ID Group ID is a ID associated with the particular Group of items in the Tizen Store Seller Office site. You need to register your Group Id on the Tizen Store Seller Office website first.

Key	Value	Description
_mcc	MCC Code, such as 250	MCC Code This information is not mandatory. It can be used only in developer mode. You can receive a list of available MCC codes through GetCountryList operation
_mnc	MNC Code, such as 01	MNC Code This information is not mandatory. It can be used only in developer mode. MNC Code is Mobile Network Code.

Example code for retrieving a list of purchased items

```
HashMappArgList;
pArgList.Construct();

String key_mode = L"_mode";
String val_mode = L"0";

String key_transactionId = L"_transactionId";
String val_transactionId = L"1";

String key_startNumber = L"_startNumber";
String val_startNumber = L"1";

String key_endNumber = L"_endNumber";
String val_endNumber = L"15";

String key_itemGroupId = L"_itemGroupId";
String val_itemGroupId = L"100000001455";

pArgList.Add(&key_mode, &val_mode);
pArgList.Add(&key_transactionId, &val_transactionId);
pArgList.Add(&key_startNumber, &val_startNumber);
pArgList.Add(&key_endNumber, &val_endNumber);
pArgList.Add(&key_itemGroupId, &val_itemGroupId);

AppManager* pAppManager = AppManager::GetInstance();
AppControl* pAc = pAppManager->FindAppControlN(L"tizeninapp.IapService",
      L"http://tizen.org/appcontrol/operation/iap/get_purchased_item_list");

pAc->Start(null, null, &pArgList, this);

delete pAc;
```

Output Data

The results of the operation are returned in the `OnAppControlCompleteResponseReceived()` event handler.

The following table shows the output data for the `http://tizen.org/appcontrol/operation/iap/get_purchased_item_list` operation:

Key	Value	Description
`_method`	`OnPurchasedItem InformationListReceived`	Method to be called as a GetPurchasedItemList request result. This value will be `OnPurchasedItem InformationListReceived`
`_result`	`<Result code>`	Result code number. Error codes are described in the end of this section.
`_transactionId`	`<TransactionID>`	Transaction ID number This ID will be the same as a Transaction ID used during request.
`_startNumber`	`<start number>`	Start Number Index of the first item on the list
`_endNumber`	`<end number>`	End Number Index of the last item on the list
`_totalCount`	`<total count>`	Total items count Number of items based on the startNumber and endNumber

We also have _totaCount number of items in the output data, each key is a PREFIX (list item index value) plus a key (i.e. “12_itemId”) as below:

Key	Value	Description
`PREFIX_itemId`	`<ItemID>`	Item ID number This ID will be the same as a Item ID used during request.

Key	Value	Description
PREFIX_itemGroupId	<ItemGroup ID>	Item group ID Item Group ID is a collection of items and it will be linked to your application in Tizen seller site.
PREFIX_itemName	<Item Name>	Item Name Item Name is a name provided during item registration on the Tizen Store Seller Office.
PREFIX_currencyUnit	Currency unit, such as: $,Won, Pound	Currency Unit Device user currency unit.
PREFIX_unitPrecedes	0 | 1	Unit Precedes String representation of values: 0 : Tall (2.99 TL) 1 : Front ($ 2.99)
PREFIX_hasPenny	0 | 1	Has Penny Informs if currency unit has penny: representation 0 : no 1 : yes
PREFIX_itemPrice	<Item price>	Item price This is a price of item in local currency.
PREFIX_itemDownloadUrl	<URL>	Item download URL This is a URL provided during item registration on the Tizen Store Seller Office.
PREFIX_itemImageUrl	<URL>	Item image URL This is a URL provided during item registration on the Tizen Store Seller Office

Key	Value	Description
PREFIX_itemDescription	<Item description>	Item description A description provided during item registration.
PREFIX_reserved1	<Reserved Field 1>	Reserved Field 1
PREFIX_reserved2	<Reserved Field 2>	Reserved Field 2
PREFIX_paymentId	<Payment ID>	ID of payment
PREFIX_purchaseDate	<Date>	Date of purchase

Example code for retrieving a list of purchased items

```
void BuyEngine::OnAppControlCompleteResponseReceived(const AppId&appId, const String&operationId,
    AppCtrlResultappControlResult, const Collection::IMap*pResultMap)
{
if(pResultMap->GetValue(String(L"_method"))!= null)
{
        String start = *((String*)(pResultMap->GetValue(String(L"_startNumber"))));
        String end = *((String*)(pResultMap->GetValue(String(L"_endNumber"))));
        intstartNum, endNum;
        Integer::Parse(start, startNum);
        Integer::Parse(start, endNum);
        for (int i=startNum; i<=endNum; ++i)
    {
        String prefix;
        prefix.Append(i);

        String key_itemId = prefix;
        key_itemId.Append(L"_itemId");
        String itemId = *((String*)(pResultMap->GetValue(key_itemId)));

        String key_itemName = prefix;
        key_itemName.Append(L"_itemName");
        String itemName = *((String*)(pResultMap->GetValue(key_itemName)));

        . . .
    }
}
}
```

GetCountryList operation

This operation returns a list of countries available for testing in-application purchase.

Input data

The following table show the (key, value) pairs required in the input extra data for the `http://tizen.org/appcontrol/operation/iap/get_country_list` operation:

Key	Value	Description
`_transacionId`	Transaction ID number, such as `1, 2, …`	Transaction ID. This information is mandatory. This is used to track a transaction between requests.

Example code for GetItemList

```
HashMappArgList;
pArgList.Construct();

String key_transactionId = L"_transactionId";
String val_transactionId = L"1";

pArgList.Add(&key_transactionId, &val_transactionId);

AppManager* pAppManager = AppManager::GetInstance();
AppControl* pAc = pAppManager->FindAppControlN(L"tizeninapp.IapService",
      L"http://tizen.org/appcontrol/operation/iap/get_country_list");

pAc->Start(null, null, &pArgList, this);

delete pAc;
```

Output Data

The results of the operation are returned in the `OnAppControlCompleteResponseReceived()` event handler.

The following table shows the output data for the `http://tizen.org/appcontrol/operation/iap/get_country_list` operation:

Key	Value	Description
`_method`	`OnCountryListReceived`	Method to be called as a purchase request result. This value will be `OnCountryListReceived`
`_result`	`<Result code>`	Result code number. Error codes are described in the end of this section.

Key	Value	Description
_transactionId	<TransactionID>	Transaction ID number This ID will be the same as a Transaction ID used during request.
_startNumber	<start number>	Start Number Index of the first item on the list
_endNumber	<end number>	End Number Index of the last item on the list
_totalCount	<total count>	Total items count Number of items based on the startNumber and endNumber

We also have _totaCount number of countries in the output data, each key is a PREFIX (list item index value) plus a key (i.e. “1_countryName”) as below:

Key	Value	Description
PREFIX_countryName	<String>	Country Name Name of a country.
PREFIX_mcc	<String>	MCC Mobile country code.

Result code values

Following is a list of possible values of `_result` key for the `tizeninapp.IapService` operations:

Value	Description	Remarks
-1	*Unknown*	The status code for unknown status
0	*Succeed*	The status code for success
100	*Cancel*	The status code if the user cancels
200	*NetworkError*	The status code for network errors
9000	*ProcessError*	The status code for process errors
9200	*ServiceUnavailable*	The status code for service errors
9201	*ItemGroupIdNotFound*	The status code if the item group ID is not found
9203	*PaymentIdNotFound*	The status code if the payment ID is not found

9207	*ItemIdNotFound*	The status code if the item ID is not found
9107	*IInvalidIemGroupId*	The status code if the item group ID is invalid
9100	*WrongNumberOfParam*	The status code if the parameter passed are wrong
1000	*WrongEndNumber*	The status code if the wrong parameter passed **(invalid endNum)**

3.3 IAP Client Control (Purchase of item)

The IAP Client **Tizen::App::AppControl** instance allows you to initialize an purchase of item and to complete the purchase.

Application ID

This application control can be accessed using aliased application ID of "`tizeninapp.IapClient`".

Operation ID

This application supports the `http://tizen.org/appcontrol/operation/iap/purchase` operation only.

Purchase operation (initialization)

This operation launches the purchase application and allows purchasing of In-App items. The input data passed in this operation are used to display a purchase form for particular item. The device user can choose between available payment methods, register his credit card, and confirm the purchase. During initialization operation no charges will be made on the device user account.

Input data

The following table show the (key, value) pairs required in the input extra data for the `http://tizen.org/appcontrol/operation/iap/purchase` operation.

Key	Value	Description
`_mode`	`0 | 1`	Mode type. This information is optional. The default value is `0` `0` is normal (commercial) mode. This mode need to be used in application submitted to Tizen Store. `1` is developer mode (always success) This mode can be used while development of the application for testing purpose. Payment will always succeed.

Key	Value	Description
_transacionId	Transaction ID number, such as 1, 2, …	Transaction ID. This information is mandatory. This is used to track a transaction between requests.
_itemId	ID of Item, such as 000000003501	Item ID This information is mandatory. Item ID is an ID associated with the particular Item in the Tizen Store Seller Office site. You can also retrieve a list of items available for purchase through tizeninapp.IapService Application ID
_itemGroupId	ID of Group, such as 100000001455	Group ID Group ID is a ID associated with the particular Group of items in the Tizen Store Seller Office site. You need to register your Group Id on the Tizen Store Seller Office website first.
_mcc	MCC Code, such as 250	MCC Code This information is not mandatory. It can be used only in developer mode. You can receive a list of available MCC codes through GetCountryList operation
_mnc	MNC Code, such as 01	MNC Code This information is not mandatory. It can be used only in developer mode. MNC Code is Mobile Network Code.

Example code for Purchase initialization

```
HashMappArgList;
pArgList.Construct();

String key_mode = L"_mode";
String val_mode = L"0";

String key_transactionId = L"_transactionId";
String val_transactionId = L"1";

String key_itemId = L"_itemId";
```

```
String val_itemId = L"000000003501";

String key_itemGroupId = L"_itemGroupId";
String val_itemGroupId = L"100000001455";

pArgList.Add(&key_mode, &val_mode);
pArgList.Add(&key_transactionId, &val_transactionId);
pArgList.Add(&key_itemId, &val_itemId);
pArgList.Add(&key_itemGroupId, &val_itemGroupId);

AppManager* pAppManager = AppManager::GetInstance();
AppControl* pAc = pAppManager->FindAppControlN(L"tizeninapp.IapClient",
L"http://tizen.org/appcontrol/operation/iap/purchase");

pAc->Start(null, null, &pArgList, this);

deletepAc;
```

Output Data

The results of the operation are returned in the `OnAppControlCompleteResponseReceived()` event handler.

This data can be used to verify the payment with Tizen Store IAP Server.

The following table shows the output data for the `http://tizen.org/appcontrol/operation/iap/purchase` operation during initialization of purchase:

Key	Value	description
`_method`	`OnPurchaseItemInitialized | OnPurchaseItemFinished`	Method to be called as a purchase request result During initialization `OnPurchaseItemInitialized` value will be returned.
`_result`	`<Result code>`	Result code number. Error codes are described in the end of this section.
`_transactionId`	`<TransactionID>`	Transaction ID number This ID will be the same as a Transaction ID used during request.
`_itemId`	`<ItemID>`	Item ID number This ID will be the same as a Item ID used during request.
`_ticketPurchaseId`	`<Purchase Ticked ID>`	Purchase ticked ID

Key	Value	description
		This ID can be used to verify the purchase with Tizen Store IAP Server.
`_ticketVerifyUrl`	URL, such as: `http://tizenstore.com`	Server's URL This URL can be used with combination of other parameters to verify the purchase with Tizen Store IAP Server.
`_ticketParam1`	`<Ticket parameter 1>`	Parameter 1 This parameter is to be used with URL
`_ticketParam2`	`<Ticket parameter 2>`	Parameter 2 This parameter is to be used with URL
`_ticketParam3`	`<Ticket parameter 3>`	Parameter 3 This parameter is to be used with URL

Purchase operation (finalization)

This operation is used to finalize the purchase.

Note: This operation is need to be launched when `OnAppControlCompleteResponseReceived()` is received from the purchase initialization operation.

Depends on whether the `developer mode` is on or not Tizen Store IAP Server will try to charge device user account with the payment.

Input data

The following table show the (key, value) pairs required in the input extra data for the `http://tizen.org/appcontrol/operation/iap/purchase` operation.

Key	Value	Description
`_transacionId`	Transaction ID number, such as `1, 2, …`	Transaction ID. This information is mandatory. This is used to track a transaction between requests.
`_purchaseResume`	`1 | 0`	Resume This information is mandatory. Resume should be set to `1` to complete the purchase.

Request example:

```
void BuyEngine::OnAppControlCompleteResponseReceived(const AppId &appId, const String &operationId,
AppCtrlResult appControlResult, const Collection::IMap *pResultMap)
{
if(pResultMap->GetValue(String(L"_method")) != null)
{
        String method = *((String*)(pResultMap->GetValue(String(L"_method"))));
}
}
```

Output data

The results of the operation are returned in the `OnAppControlCompleteResponseReceived()` event handler.

This data can be used to verify the payment with Tizen Store IAP Server.

The following table shows the output data for the `http://tizen.org/appcontrol/operation/iap/purchase` operation during finalization of purchase:

Key	Value	description
`_method`	`OnPurchaseItemInitialized | OnPurchaseItemFinished`	Method to be called as a purchase request result During finalization `OnPurchaseItemFinalized` value will be returned.
`_result`	`<Result code>`	Result code number. Error codes are described in the end of this section.
`_transactionId`	`<TransactionID>`	Transaction ID number This ID will be the same as a Transaction ID used during request.
`_itemId`	`<ItemID>`	Item ID number This ID will be the same as a Item ID used during request.
`_itemGroupId`	`<ItemGroup ID>`	Item group ID Item Group ID is a collection of items and it will be linked to your application in Tizen seller site.

Key	Value	description
_itemName	<Item Name>	Item Name Item Name is a name provided during item registration on the Tizen Store Seller Office.
_currencyUnit	Currency unit, such as: $,Won, Pound	Currency Unit Device user currency unit.
_unitPrecedes	0 | 1	Unit Precedes String representation of values: 0 : Tall (2.99 TL) 1 : Front ($ 2.99)
_hasPenny	0 | 1	Has Penny Informs if currency unit has penny: representation 0 : no 1 : yes
_itemPrice	<Item price>	Item price This is a price of item in local currency.
_itemDownloadUrl	<URL>	Item download URL This is a URL provided during item registration on the Tizen Store Seller Office.
_itemImageUrl	<URL>	Item image URL This is a URL provided during item registration on the Tizen Store Seller Office
_itemDescription	<Item description>	Item description A description provided during item registration.
_reserved1	<Reserved Field 1>	Reserved Field 1
_reserved2	<Reserved Field 2>	Reserved Field 2
_paymentId	<Payment ID>	Payment ID ID of payment.

Key	Value	description
`_purchaseDate`	`Date`	Purchase date Date of purchase.

Result code values

Following is a list of possible values of `_result` key of the `http://tizen.org/appcontrol/operation/iap/purchase` operation.

Value	Description	Remarks
-1	*Unknown*	The status code for unknown status
0	*Succeed*	The status code for success
100	*Cancel*	The status code if the user cancels
200	*NetworkError*	The status code for network errors
9000	*ProcessError*	The status code for process errors
9200	*ServiceUnavailable*	The status code for service errors
9201	*ItemGroupIdNotFound*	The status code if the item group ID is not found
9203	*PaymentIdNotFound*	The status code if the payment ID is not found
9207	*ItemIdNotFound*	The status code if the item ID is not found
1000	*InvalidParameter*	The status code if the parameter is invalid

4. Verifying with Tizen Store IAP Server

After `http://tizen.org/appcontrol/operation/iap/purchase` operation is successfully initialized, you can use output data's, values to verify a purchase on the Tizen Store IAP Server. It's simple process. You can send a query to output data's `_ticketVerifyUrl` address. The query can be made with combination of output data's `_ticketPurchaseId`, `_ticketParam1`, `_ticketParam2`, `_ticketParam3`.

The below is an example of request:

```
http://iap.tizenstore.com/appsItemVerifyIAPReceipt.as?purchaseID=2fbdf6f763d2f3fb8de648
3f71258ca44d0c2ef18812168eb54bf9e5e1d7ef96&param1=2ef5b6b79e76a4067f547abc5835512408b0a
b39b1ae190f270ce85e2a7b24ce&param2=f2e786faf99324baa7ee2642f0323135c711fcbf12d2b28a0363
a37d091d69bd&param3=6d00026c849852fd98879e019276602a80a121ddb3693d6affff2f489b85e1e
```

If a request is processed successfully, the JSON response is received as shown. The status value of "true" means a verifying the purchase is successful and 'false' when it has failed.

The JSON example of successful result:

```
{"paymentID":"TPMTID20130607RU00000240","paymentAmount":"35.0","itemName":"Item
01","itemID":"000000003501","status":"true","purchaseDate":"2013-06-07
11:35:29","itemDesc":"Item 01"}
```

The JSON example of failed result

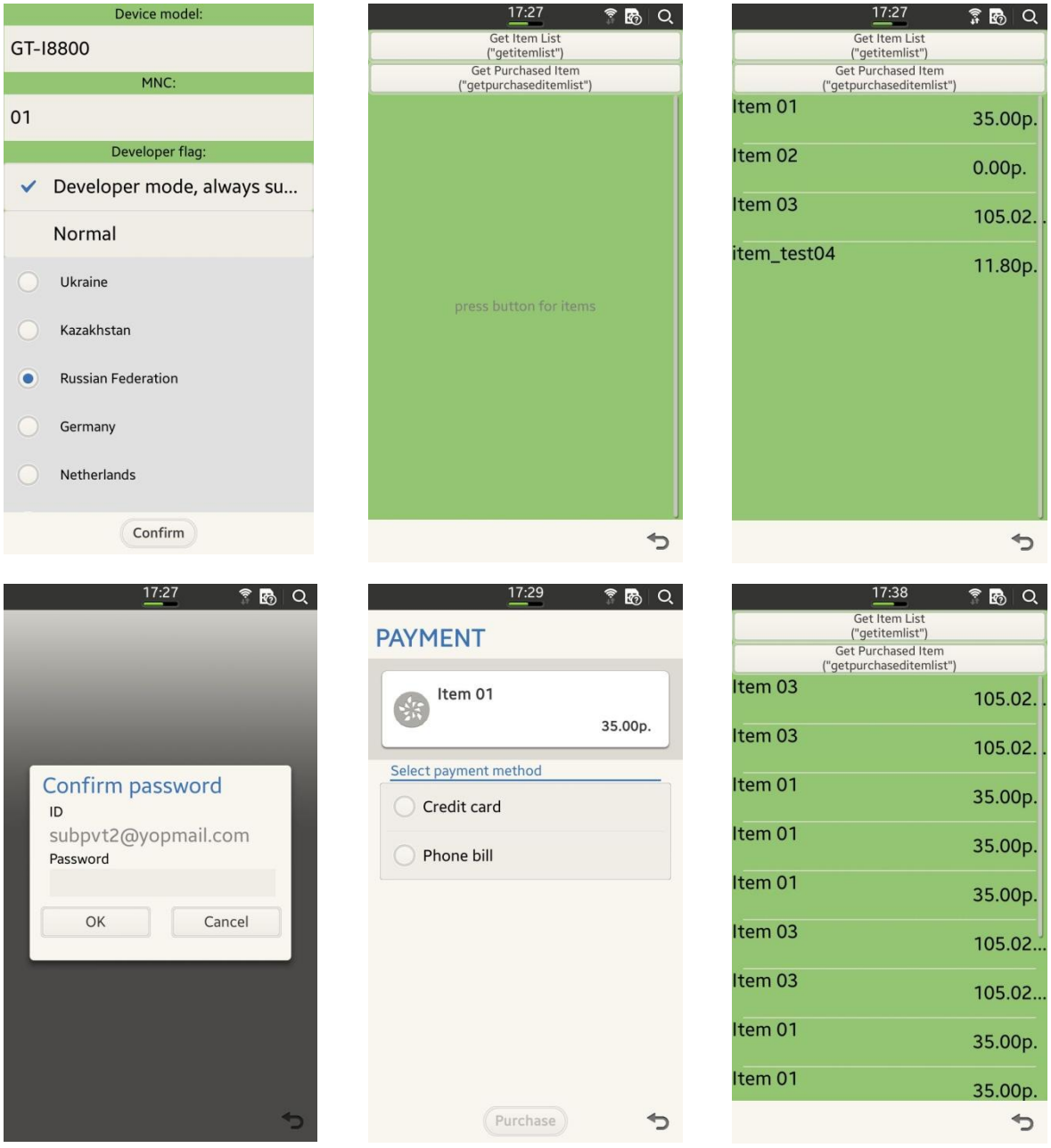
```
{"status":"false"}
```

If your application uses a server-client model, server-to-server purchase verification explained above is recommended.

Appendixes

Attached sample application

The attached sample application allows a user to show a list of items for purchase, purchase an item, show purchased items, and make a purchase:



The sample flow of purchase item

Above is an example of third case: when a user clicks on the “Simple purchase” button, a list of items will be shown, and a user will be able to purchase an item.