

Consumer Mobile SDK

Reference Guide

Version 0.1

# Document Configuration Management

## Document Identification

|  |  |
| --- | --- |
| **Name of Project** |  |
| **Name of Report** | Consumer Mobile SDK Reference Guide |
| **Version** | 0.1 (draft) |
| **File Name** |  |
| **Sensitivity Classification** | Confidential |

## Document Change Control

|  |  |  |
| --- | --- | --- |
| Version | Date Released | Change Notice |
| 0.1 |  | Initial (draft) for review and feedback |
|  |  |  |

Table of Contents

[Document Configuration Management 2](#_Toc20154639)

[Document Identification 2](#_Toc20154640)

[Document Preparation 2](#_Toc20154641)

[Document Change Control 2](#_Toc20154642)

[1 Introduction 12](#_Toc20154643)

[12](#_Toc20154644)

[2 Init 13](#_Toc20154645)

[2.1 ECSServiceProvider 13](#_Toc20154646)

[Parameters 13](#_Toc20154647)

[Output 13](#_Toc20154648)

[2.2 Sample 13](#_Toc20154649)

[3 Set Proposition ID 13](#_Toc20154650)

[3.1 ECSServiceProvider 13](#_Toc20154651)

[Parameters 13](#_Toc20154652)

[3.2 Sample 13](#_Toc20154653)

[4 Configure ECS 14](#_Toc20154654)

[4.1 ECSServiceProvider 14](#_Toc20154655)

[Parameters 14](#_Toc20154656)

[Output 14](#_Toc20154657)

[4.2 Sample 14](#_Toc20154658)

[4.3 Errors 14](#_Toc20154659)

[5 Configure ECS to Fetch Configuration 15](#_Toc20154660)

[5.1 ECSServiceProvider 15](#_Toc20154661)

[Parameters 15](#_Toc20154662)

[Output 15](#_Toc20154663)

[5.2 Sample 15](#_Toc20154664)

[5.3 Errors 15](#_Toc20154665)

[6 Hybris OAuth Authentication 15](#_Toc20154666)

[6.1 ECSServiceProvider 16](#_Toc20154667)

[Parameters 16](#_Toc20154668)

[Output 16](#_Toc20154669)

[6.2 Sample 16](#_Toc20154670)

[6.3 Errors 16](#_Toc20154671)

[7 Refresh Hybris OAuth 17](#_Toc20154672)

[7.1 ECSServiceProvider 17](#_Toc20154673)

[Parameters 17](#_Toc20154674)

[Output 17](#_Toc20154675)

[7.2 Sample 18](#_Toc20154676)

[7.3 Errors 18](#_Toc20154677)

[8 Fetch Product List 18](#_Toc20154678)

[8.1 ECSServiceProvider 18](#_Toc20154679)

[Parameters 19](#_Toc20154680)

[Output 19](#_Toc20154681)

[8.2 Sample 19](#_Toc20154682)

[8.3 Errors 19](#_Toc20154683)

[9 Fetch Product for CTN 20](#_Toc20154684)

[9.1 ECSServiceProvider 20](#_Toc20154685)

[Parameters 20](#_Toc20154686)

[Output 20](#_Toc20154687)

[9.2 Sample 20](#_Toc20154688)

[9.3 Errors 20](#_Toc20154689)

[10 Fetch Product List for Set of CTNs 22](#_Toc20154690)

[10.1 ECSServiceProvider 22](#_Toc20154691)

[Parameters 22](#_Toc20154692)

[Output 22](#_Toc20154693)

[10.2 Sample 22](#_Toc20154694)

[10.3 Errors 22](#_Toc20154695)

[11 Fetch Product Details 24](#_Toc20154696)

[11.1 ECSServiceProvider 24](#_Toc20154697)

[Parameters 24](#_Toc20154698)

[Output 24](#_Toc20154699)

[11.2 Sample 24](#_Toc20154700)

[11.3 Errors 24](#_Toc20154701)

[12 Create Shopping Cart 25](#_Toc20154702)

[12.1 ECSServiceProvider 25](#_Toc20154703)

[Parameters 25](#_Toc20154704)

[Output 25](#_Toc20154705)

[12.2 Sample 25](#_Toc20154706)

[12.3 Errors 26](#_Toc20154707)

[13 Add Product to Shopping Cart 26](#_Toc20154708)

[13.1 ECSServiceProvider 26](#_Toc20154709)

[Parameters 26](#_Toc20154710)

[Output 26](#_Toc20154711)

[13.2 Sample 27](#_Toc20154712)

[13.3 Errors 27](#_Toc20154713)

[14 Fetch Shopping Cart 27](#_Toc20154714)

[14.1 ECSServiceProvider 27](#_Toc20154715)

[Parameters 27](#_Toc20154716)

[Output 28](#_Toc20154717)

[14.2 Sample 28](#_Toc20154718)

[14.3 Errors 28](#_Toc20154719)

[15 Update Shopping Cart 29](#_Toc20154720)

[15.1 ECSServiceProvider 29](#_Toc20154721)

[Parameters 29](#_Toc20154722)

[Output 29](#_Toc20154723)

[15.2 Sample 29](#_Toc20154724)

[15.3 Errors 29](#_Toc20154725)

[16 Apply Voucher 30](#_Toc20154726)

[16.1 ECSServiceProvider 30](#_Toc20154727)

[Parameters 30](#_Toc20154728)

[Output 30](#_Toc20154729)

[16.2 Sample 30](#_Toc20154730)

[16.3 Errors 31](#_Toc20154731)

[17 Fetch Applied Voucher 31](#_Toc20154732)

[17.1 ECSServiceProvider 31](#_Toc20154733)

[Parameters 31](#_Toc20154734)

[Output 31](#_Toc20154735)

[17.2 Sample 32](#_Toc20154736)

[17.3 Errors 32](#_Toc20154737)

[18 Remove Voucher 33](#_Toc20154738)

[18.1 ECSServiceProvider 33](#_Toc20154739)

[Parameters 33](#_Toc20154740)

[Output 33](#_Toc20154741)

[18.2 Sample 33](#_Toc20154742)

[18.3 Errors 33](#_Toc20154743)

[19 Set Delivery Mode 35](#_Toc20154744)

[19.1 ECSServiceProvider 35](#_Toc20154745)

[Parameters 35](#_Toc20154746)

[Output 35](#_Toc20154747)

[19.2 Sample 35](#_Toc20154748)

[19.3 Errors 35](#_Toc20154749)

[20 Fetch Delivery Modes 36](#_Toc20154750)

[20.1 ECSServiceProvider 36](#_Toc20154751)

[Parameters 36](#_Toc20154752)

[Output 36](#_Toc20154753)

[20.2 Sample 36](#_Toc20154754)

[20.3 Errors 37](#_Toc20154755)

[21 Fetch Regions 37](#_Toc20154756)

[21.1 ECSServiceProvider 37](#_Toc20154757)

[Parameters 37](#_Toc20154758)

[Output 37](#_Toc20154759)

[21.2 Sample 37](#_Toc20154760)

[21.3 Errors 38](#_Toc20154761)

[22 Create Address 38](#_Toc20154762)

[22.1 ECSServiceProvider 38](#_Toc20154763)

[Parameters 38](#_Toc20154764)

[Output 38](#_Toc20154765)

[22.2 Sample 39](#_Toc20154766)

[22.3 Errors 39](#_Toc20154767)

[23 Create and Fetch Address 39](#_Toc20154768)

[23.1 ECSServiceProvider 39](#_Toc20154769)

[Parameters 40](#_Toc20154770)

[Output 40](#_Toc20154771)

[23.2 Sample 40](#_Toc20154772)

[23.3 Errors 40](#_Toc20154773)

[24 Fetch Saved Address 41](#_Toc20154774)

[24.1 ECSServiceProvider 41](#_Toc20154775)

[Parameters 41](#_Toc20154776)

[Output 41](#_Toc20154777)

[24.2 Sample 41](#_Toc20154778)

[24.3 Errors 41](#_Toc20154779)

[25 Set Delivery Address 42](#_Toc20154780)

[25.1 ECSServiceProvider 42](#_Toc20154781)

[Parameters 42](#_Toc20154782)

[Output 42](#_Toc20154783)

[25.2 Sample 42](#_Toc20154784)

[25.3 Errors 42](#_Toc20154785)

[26 Set and Fetch Delivery Address 43](#_Toc20154786)

[26.1 ECSServiceProvider 43](#_Toc20154787)

[Parameters 43](#_Toc20154788)

[Output 43](#_Toc20154789)

[26.2 Sample 43](#_Toc20154790)

[26.3 Errors 44](#_Toc20154791)

[27 Update Address 44](#_Toc20154792)

[27.1 ECSServiceProvider 44](#_Toc20154793)

[Parameters 44](#_Toc20154794)

[Output 44](#_Toc20154795)

[27.2 Sample 45](#_Toc20154796)

[27.3 Errors 45](#_Toc20154797)

[28 Update and Fetch Address 45](#_Toc20154798)

[28.1 ECSServiceProvider 46](#_Toc20154799)

[Parameters 46](#_Toc20154800)

[Output 46](#_Toc20154801)

[28.2 Sample 46](#_Toc20154802)

[28.3 Errors 46](#_Toc20154803)

[29 Delete Address 47](#_Toc20154804)

[29.1 ECSServiceProvider 47](#_Toc20154805)

[Parameters 47](#_Toc20154806)

[Output 47](#_Toc20154807)

[29.2 Sample 47](#_Toc20154808)

[29.3 Errors 48](#_Toc20154809)

[30 Delete and Fetch Address 48](#_Toc20154810)

[30.1 ECSServiceProvider 48](#_Toc20154811)

[Parameters 48](#_Toc20154812)

[Output 48](#_Toc20154813)

[30.2 Sample 49](#_Toc20154814)

[30.3 Errors 49](#_Toc20154815)

[31 Fetch Retailers 49](#_Toc20154816)

[31.1 ECSServiceProvider 49](#_Toc20154817)

[Parameters 49](#_Toc20154818)

[Output 49](#_Toc20154819)

[31.2 Sample 50](#_Toc20154820)

[31.3 Errors 50](#_Toc20154821)

[32 Fetch Retailers 50](#_Toc20154822)

[32.1 ECSServiceProvider 50](#_Toc20154823)

[Parameters 50](#_Toc20154824)

[Output 51](#_Toc20154825)

[32.2 Sample 51](#_Toc20154826)

[32.3 Errors 51](#_Toc20154827)

[33 Set Payment Details 51](#_Toc20154828)

[33.1 ECSServiceProvider 51](#_Toc20154829)

[Parameters 52](#_Toc20154830)

[Output 52](#_Toc20154831)

[33.2 Sample 52](#_Toc20154832)

[33.3 Errors 52](#_Toc20154833)

[34 Fetch Payment Details 53](#_Toc20154834)

[34.1 ECSServiceProvider 53](#_Toc20154835)

[Parameters 53](#_Toc20154836)

[Output 53](#_Toc20154837)

[34.2 Sample 53](#_Toc20154838)

[34.3 Errors 53](#_Toc20154839)

[35 Submit Order 54](#_Toc20154840)

[35.1 ECSServiceProvider 54](#_Toc20154841)

[Parameters 54](#_Toc20154842)

[Output 54](#_Toc20154843)

[35.2 Sample 54](#_Toc20154844)

[35.3 Errors 54](#_Toc20154845)

[36 Make Payment 55](#_Toc20154846)

[36.1 ECSServiceProvider 55](#_Toc20154847)

[Parameters 55](#_Toc20154848)

[Output 55](#_Toc20154849)

[36.2 Sample 55](#_Toc20154850)

[36.3 Errors 56](#_Toc20154851)

[37 Fetch Order History 56](#_Toc20154852)

[37.1 ECSServiceProvider 56](#_Toc20154853)

[Parameters 56](#_Toc20154854)

[Output 56](#_Toc20154855)

[37.2 Sample 57](#_Toc20154856)

[37.3 Errors 57](#_Toc20154857)

[38 Fetch Order Detail 57](#_Toc20154858)

[38.1 ECSServiceProvider 57](#_Toc20154859)

[Parameters 57](#_Toc20154860)

[Output 57](#_Toc20154861)

[38.2 Sample 58](#_Toc20154862)

[38.3 Errors 58](#_Toc20154863)

[39 Fetch Order Detail 58](#_Toc20154864)

[39.1 ECSServiceProvider 58](#_Toc20154865)

[Parameters 58](#_Toc20154866)

[Output 59](#_Toc20154867)

[39.2 Sample 59](#_Toc20154868)

[39.3 Errors 59](#_Toc20154869)

[40 Fetch Order Detail 59](#_Toc20154870)

[40.1 ECSServiceProvider 59](#_Toc20154871)

[Parameters 60](#_Toc20154872)

[Output 60](#_Toc20154873)

[40.2 Sample 60](#_Toc20154874)

[40.3 Errors 60](#_Toc20154875)

[41 Fetch User Profile 60](#_Toc20154876)

[41.1 ECSServiceProvider 61](#_Toc20154877)

[Parameters 61](#_Toc20154878)

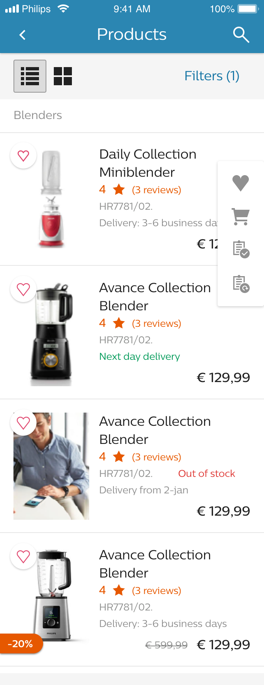
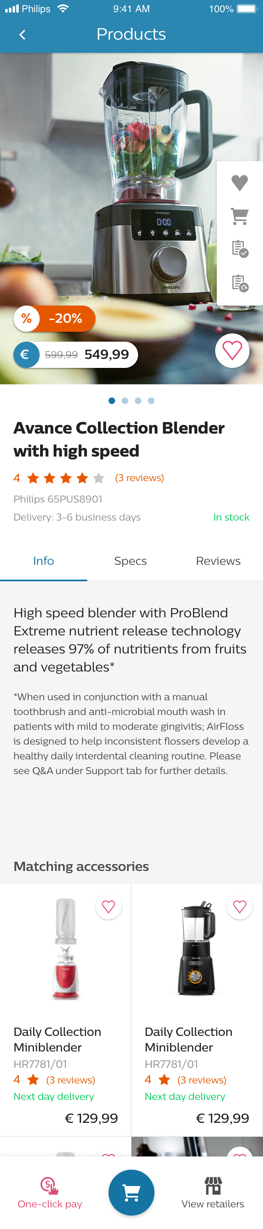
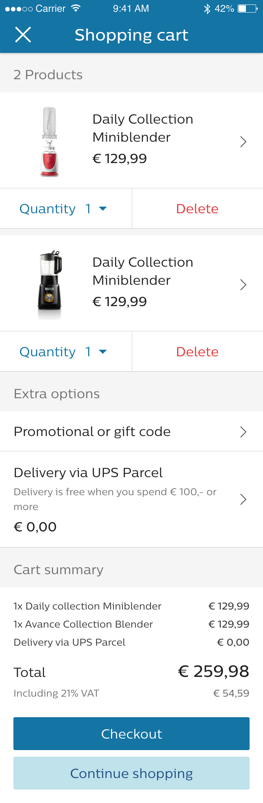
[Output 61](#_Toc20154879)

[41.2 Sample 61](#_Toc20154880)

[41.3 Errors 61](#_Toc20154881)

# **Introduction**

# 



The e-commerce software development kit (SDK) defines the In-App purchase experience and how it is delivered to the users. Embedding this into your existing android application enables you to design an integrated solution.

The SDK provides the necessary tools to build the application and gives you control over how the application looks and operates. Your own workflow and images can be used to design all aspects of user experience such as how product list appears, how to display shopping cart, displaying address, what details to be shown and what information to collect or not collect during the intake process. This ability to uniquely tailor In-App capabilities offers your application limitless possibilities for using the SDK and gives good user experience.

# **Init**

This method is used to initialize the ECSSDKService.

## ECSServiceProvider

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

**public** ECSServices(@Nullable String propositionID, @NonNull appInfra)

{  
   
}

## Parameters

* propositionID – It is a unique store identification string ,which is specific to business and country
* appInfra – appInfra object created by proposition

## Output

* ECSSDKService object

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);

# **Set Proposition ID**

This method is used to set proposition id.

## ECSServiceProvider

**void** setPropositionID(@NonNull String propositionID);

## Parameters

* propositionID – It is a unique store identification string ,which is specific to business and country

## Sample

ECSConfiguration.***INSTANCE***.setPropositionID(**"your propositionID"**);

# **Configure ECS**

This method is used to configure the ECS.

## ECSServiceProvider

**void** configureECS(ECSCallback<Boolean,Exception> ecsCallback);

## Parameters

* ecsCallback – instance of ECSCallBack of type Boolean and Exception.

## Output

* Success response – If callback is success, returns true else false

True : represents Hybris flow

False : represents Retailer flow

* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.configureECS(**new** ECSCallback<Boolean, Exception>() {  
 @Override  
 **public void** onResponse(Boolean result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Server Error

* ECSSomethingWentWrong (We have encountered technical glitch. Please try after some time) – This error occurs when failed to get response or the proposition id might not be configured in Hybris

# **Configure ECS to Fetch Configuration**

This method is used to fetch configuration from Hybris for the specific proposition id.

## ECSServiceProvider

**void** configureECSToGetConfiguration(ECSCallback<ECSConfig, Exception> ecsCallback);

## Parameters

* ecsCallback – instance of ECSCallback of type ECSConfig and Exception

## Output

* Success response – ECSConfig object which contains all the ECS configuration data
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.configureECSToGetConfiguration(**new** ECSCallback<ECSConfig, Exception>() {  
 @Override  
 **public void** onResponse(ECSConfig result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSsomethingWentWrong (We have encountered technical glitch. Please try after some time) – This error occurs when failed to get response.

# **Hybris OAuth Authentication**

This method is used to make the OAuth authentication with Hybris. User needs to send the object of class ECSOAuthProvider which contains all the authentication related data as a parameter to the method.

## ECSServiceProvider

**void** hybrisOAthAuthentication(@NonNull ECSOAuthProvider ecsoAuthProvider, @NonNull ECSCallback<ECSOAuthData,Exception> ecsListener);

## Parameters

* ecsoAuthProvider – object of the class ECSOAuthProvider which provide all oauth related data
* ecsCallback – instance of ECSCallback of type ECSOAuthData and Exception

## Output

* Success response – ECSOAuthData object which contains Janrain id
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);

ECSOAuthProvider oAuthInput = **new** ECSOAuthProvider() {  
 @Override  
 publicString getOAuthID() {  
 return "your Jainrain ID**"**;  
 }  
};

ecsServices. hybrisOAthAuthentication(oAuthInput, **new** ECSCallback<ECSOAuthData, Exception>() {  
 @Override  
 **public void** onResponse(ECSOAuthData result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSOAuthDetailError (Please provide complete OAuth details) – This error occurs when any OAuth detail is missing, so provide all the required details
* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale
* ECSInvalidTokenError (We have encountered technical glitch (Invalid access token))

Server Errors

* ECSinvalid\_grant (We have encountered technical glitch (Invalid Janrain Token). Please re-login) – This error occurs when the Janrain token is invalid
* ECSinvalid\_client (We have encountered technical glitch (Bad client credentials). Please re-login)

# **Refresh Hybris OAuth**

This method is used to refresh the OAuth authentication with Hybris. User needs to send the object of class ECSOAuthProvider which contains all the authentication related data as a parameter to the method.

## ECSServiceProvider

**void** hybrisRefreshOAuth(ECSOAuthProvider oAuthInput, ECSCallback<ECSOAuthData, Exception> ecsListener);

## Parameters

* ecsoAuthProvider – object of the class ECSOAuthProvider which provide all oauth related data
* ecsCallback – instance of ECSCallback of type ECSOAuthData and Exception

## Output

* Success response – ECSOAuthData object which contains refresh token
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);

ECSOAuthProvider oAuthInput = **new** ECSOAuthProvider() {  
 @Override  
 publicString getOAuthID() {  
 return "your refresh token**"**;  
 }  
};

ecsServices. hybrisRefreshOAuth(oAuthInput, **new** ECSCallback<ECSOAuthData,Exception>() {  
 @Override  
 **public void** onResponse(ECSOAuthData result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSOAuthDetailError (Please provide complete OAuth details) – This error occurs when any OAuth detail is missing, so provide all the required details
* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSinvalid\_grant (We have encountered technical glitch (Invalid Janrain Token). Please re-login) – This error occurs when the Janrain token is invalid

# **Fetch Product List**

This method is used to fetch list of products for Hybris flow. It returns list of products having summary data fetched from PRX server. This can be called for an anonymous user as well.

## ECSServiceProvider

**void** fetchProducts(**int** currentPage, **int** pageSize,ECSCallback<ECSProducts,Exception> eCSCallback);

## Parameters

* currentPage – refer to the page number for pagination
* pageSize – refer to the total number of data per page
* eCSCallback – instance of ECSCallback of type ECSProducts (which contains list of product ) and Exception

## Output

* Success response – List of products having summary data fetched from PRX
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.fetchProducts(0, 20, **new** ECSCallback<ECSProducts, Exception>() {  
 @Override  
 **public void** onResponse(ECSProducts result) {  
  
 }  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSNoSuchElementError (An error occurred while fetching the data from the server). Please re-login)

# **Fetch Product for CTN**

This method is used to fetch product specific to ctn for Hybris flow. It returns a product having detail and summary data fetched from PRX server. This method can be called for an anonymous user as well.

## ECSServiceProvider

**void** fetchProduct(**String** ctn,ECSCallback<ECSProduct,Exception> eCSCallback);

## Parameters

* ctn – refers to the unique number of each product
* eCSCallback – instance of ECSCallback of type ECSProduct and Exception

## Output

* Success response – Product having details and summary data fetched from PRX
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);

ecsServices.fetchProduct(mCTNValue, **new** ECSCallback <ECSProduct, Exception>() {  
 @Override  
 **public void** onResponse(ECSProducts result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
 });

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale
* ECSCtnNotProvided (Please provide the CTN) – occurs when ctn or not provided or invalid ctn is entered

Server Errors

* ECSNoSuchElementError (An error occurred while fetching the data from the server). Please re-login)

# **Fetch Product List for Set of CTNs**

This method is used to fetch list of products for Hybris flow. It returns list of products having detail and summary data fetched from PRX server. This method can be called for an anonymous user as well.

## ECSServiceProvider

**void** fetchProductSummaries(**List<String>** ctns,ECSCallback<List<ECSProduct>,Exception> eCSCallback);

## Parameters

* ctns – refer to the list of unique numbers of each product
* eCSCallback – instance of ECSCallback containing list of ECSProduct and Exception

## Output

* Success response – List of products having summary data fetched from PRX
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);

ecsServices.fetchProductSummaries(productList, **new** ECSCallback<List<ECSProduct>, Exception>() {  
 @Override  
 **public void** onResponse(List<ECSProduct> result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
 });

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – occurs when locale is null, call configureECS or configureECSToGetConfiguration to set locale
* ECSCtnNotProvided (Please provide the CTN) – occurs when ctn or not provided or invalid ctn is entered

Server Errors

* ECSNoSuchElementError (An error occurred while fetching the data from the server). Please re-login)

# **Fetch Product Details**

This method is used to fetch product details containing assets and disclaimer for the specific product from PRX server.

## ECSServiceProvider

**void** fetchProductDetails(ECSProduct product, ECSCallback<ECSProduct,Exception> ecsCallback);

## Parameters

* product – refer to the object of class ECSProduct for which

detail needs to be fetched

* ecsCallback – instance of ECSCallback of type ECSProduct and Exception

## Output

* Success response – ECSProduct object having assets and disclaimer data fetched from PRX server.
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);

ecsServices.fetchProductDetails(**product**, **new** ECSCallback<ECSProduct, Exception>() {  
 @Override  
 **public void** onResponse(ECSProduct result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
 });

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale
* ECSCtnNotProvided (Please provide the CTN) – occurs when ctn is not provided or invalid ctn is entered

# **Create Shopping Cart**

This method is used to create the shopping cart for the user. User needs to oAuth before calling this method.

## ECSServiceProvider

**void** createShoppingCart(ECSCallback<ECSShoppingCart,Exception> ecsCallback);

## Parameters

* ecsCallback – instance of ECSCallback of type ECSShoppingCart and Exception

## Output

* Success response – ECSShoppingCart object
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);

ecsServices.createShoppingCart(**new** ECSCallback<ECSShoppingCart, Exception>() {  
 @Override  
 **public void** onResponse(ECSShoppingCart result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
 });

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

# **Add Product to Shopping Cart**

This method is used add product to the shopping cart.

## ECSServiceProvider

**void** addProductToShoppingCart(ECSProduct product, ECSCallback<ECSShoppingCart, Exception> ecsCallback);

## Parameters

* product – an object of type ECSProduct which needs to be

added to the shopping cart

* ecsCallback – instance of ECSCallback of type ECSShoppingCart and Exception

## Output

* Success response – ECSShoppingCart object having details of the product added to cart.
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);

ecsServices.addProductToShoppingCart(product, **new** ECSCallback<ECSShoppingCart, Exception>() {  
 @Override  
 **public void** onResponse(ECSShoppingCart result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
 });

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale
* ECSCtnNotProvided (Please provide the CTN) – occurs when ctn is not provided or invalid ctn is entered

Server Errors

* ECSCartError (No cart created yet) – This error occurs when cart is not created, to resolve call createShoppingCart
* ECSCartEntryError (The Product is not added in Cart) – occurs when failed to add product to cart

# **Fetch Shopping Cart**

This method is used to get the shopping cart details for the user. User needs to oAuth before calling this method.

## ECSServiceProvider

**void** fetchShoppingCart(ECSCallback<ECSShoppingCart,Exception> ecsCallback);

## Parameters

* ecsCallback – instance of ECSCallback containing ECSShoppingCart and Exception

## Output

* Success response – ECSShoppingCart object
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);

ecsServices.fetchShoppingCart(**new** ECSCallback<ECSShoppingCart, Exception>() {  
 @Override  
 **public void** onResponse(ECSShoppingCart result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
 });

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSNoSuchElementError (An error occurred while fetching the data from the server). Please re-login) – This error occurs when failed to get the response
* ECSCartError (No cart created yet)

# **Update Shopping Cart**

This method is used to update the quantity in the shopping cart for the product.

## ECSServiceProvider

**void** updateShoppingCart(**int** quantity, ECSEntries entriesEntity, ECSCallback<ECSShoppingCart, Exception> ecsCallback) ;

## Parameters

* quantity – an integer value which tells the quantity which needs to be updated
* ecsCallback – instance of ECSCallback of type ECSShoppingCart and Exception

## Output

* Success response – ECSEntries object which contains object of type ECSProduct for which the quantity needs to be updated
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);

ecsServices.updateShoppingCart(count, entriesEntity, **new** ECSCallback<ECSShoppingCart, Exception>() {  
 @Override  
 **public void** onResponse(ECSShoppingCart result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
 });

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale
* ECSCommerceCartModificationError (Quantity must not be less than zero) – occurs when quantity is less than zero

Server Errors

* ECSCommerceCartModificationError (Quantity must not be less than zero) – occurs when quantity is less than zero

# **Apply Voucher**

This method is used to apply voucher on the product on cart level or products level.

## ECSServiceProvider

**void** applyVoucher(String voucherCode, ECSCallback<List<ECSVoucher>, Exception> ecsCallback);

## Parameters

* voucherCode – a string value which has to be applied on cart or product
* ecsCallback – instance of ECSCallback of type ECSVoucher and Exception

## Output

* Success response – list of ECSVoucher object with updated data
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);

ecsServices.applyVoucher(voucherId, **new** ECSCallback<List<ECSVoucher>, Exception>() {  
 @Override  
 **public void** onResponse(List<ECSVoucher> result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
 });

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSUnsupportedVoucherError (Sorry, this voucher can’t be redeemed) - occurs when invalid voucher code is entered
* ECSVoucherOperationError (Sorry, this voucher can’t be redeemed)

# **Fetch Applied Voucher**

This method is used to fetch the list of applied vouchers.

## ECSServiceProvider

**void** fetchAppliedVouchers(ECSCallback<List<ECSVoucher>, Exception> ecsCallback);

## Parameters

* ecsCallback – instance of ECSCallback containing list of ECSVoucher and Exception

## Output

* Success response – List of ECSVoucher object containing list of applied vouchers
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);

ecsServices.fetchAppliedVouchers(**new** ECSCallback<List<ECSVoucher>, Exception>() {  
 @Override  
 **public void** onResponse(List<ECSVoucher> result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
 });

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSVoucherOperationError (Sorry, this voucher can’t be redeemed)
* ECSUnsupportedVoucherError (Sorry, this voucher can’t be redeemed) - occurs when invalid voucher code is entered

# **Remove Voucher**

This method is used to remove the applied voucher.

## ECSServiceProvider

**void** removeVoucher(String voucherCode, ECSCallback<List<ECSVoucher>, Exception> ecsCallback);

## Parameters

* voucherCode – string value which is a unique number of which voucher has to be removed
* ecsCallback – instance of ECSCallback containing list of ECSVoucher and Exception

## Output

* Success response – list of ECSVoucher object
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);

ecsServices.removeVoucher(voucherCode, **new** ECSCallback<List<ECSVoucher>, Exception>() {  
 @Override  
 **public void** onResponse(List<ECSVoucher> result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
 });

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale
* ECSUnsupportedVoucherError (Sorry, this voucher can’t be redeemed) - occurs when invalid voucher code is entered

Server Errors

* ECSVoucherOperationError (Sorry, this voucher can’t be redeemed)
* ECSUnsupportedVoucherError (Sorry, this voucher can’t be redeemed) - occurs when invalid voucher code is entered

# **Set Delivery Mode**

This method is used to set delivery mode for cart.

## ECSServiceProvider

**void** setDeliveryMode(ECSDeliveryMode deliveryModes, ECSCallback<Boolean, Exception> ecsCallback);

## Parameters

* deliveryModes – ECSDeliveryMode object to be set as delivery mode
* ecsCallback – instance of ECSCallback of type Boolean and Exception

## Output

* Success response – returns true if delivery mode is set successfully
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices. setDeliveryMode(deliveryMode, **new** ECSCallback<Boolean, Exception>(){  
 @Override  
 **public void** onResponse(Boolean result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale
* ECSUnsupportedDeliveryModeError (Selected delivery mode is not supported)

Server Errors

* ECSUnsupportedDeliveryModeError (Selected delivery mode is not supported)

# **Fetch Delivery Modes**

This method is used to fetch the list of delivery modes.

## ECSServiceProvider

**void** fetchDeliveryModes(ECSCallback<List<ECSDeliveryMode>, Exception> ecsCallback);

## Parameters

* ecsCallback – instance of ECSCallback containing list of ECSDeliveryMode and Exception

## Output

* Success response – list of ECSDeliveryMode object which contains list of delivery modes
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.fetchDeliveryModes(**new** ECSCallback<List<ECSDeliveryMode>, Exception>() {  
 @Override  
 **public void** onResponse(ECSDeliveryMode result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

# **Fetch Regions**

This method is used to fetch the list of regions for the selected country.

## ECSServiceProvider

**void** fetchRegions(ECSCallback<List<ECSRegion>, Exception> ecsCallback);

## Parameters

* ecsCallback – instance of ECSCallback containing list of ECSRegion and Exception

## Output

* Success response – list of ECSRegion object which contains list of regions
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.fetchRegions(**new** ECSCallback<List<ECSRegion>, Exception>() {  
 @Override  
 **public void** onResponse(List<ECSRegion> result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSCountryCodeNotGiven (Please provide valid country code)
* ECSRegionisocode (Region selected is invalid)

# **Create Address**

This method is used to create new address for the user.

## ECSServiceProvider

**void** createAddress(ECSAddress address, ECSCallback<ECSAddress, Exception> ecsCallback);

## Parameters

* address – an object of class ECSAddress which needs to be added
* ecsCallback – instance of ECSCallback of type ECSAddress and Exception

## Output

* Success response – ECSAddress object which contains address details
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.createAddress(address, **new** ECSCallback< ECSAddress, Exception>() {  
 @Override  
 **public void** onResponse(ECSAddress result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSCountryCodeNotGiven (Please provide valid country code)
* ECSRegionisocode (Region selected is invalid) –
* ECSPostalCode (ZIP code selected is invalid)
* ECSfirstName **(**Please notice that the following characters are not allowed: \") ( \* ^ % $ # @ ! + , . ? / : ; { } [ ] ` ~ =)
* ECSlastName (Please notice that the following characters are not allowed: \") ( \* ^ % $ # @ ! + , . ? / : ; { } [ ] ` ~ =)
* ECSphone1 **(**Phone number selected is invalid)
* ECSaddressId **(**Given address doesn't exist or belong to another user)
* ECSValidationError **(**Data validation failed)

# **Create and Fetch Address**

This method is used to create new address for the user and fetch the list of all saved addresses.

## ECSServiceProvider

**void** createAndFetchAddress(ECSAddress address, ECSCallback<List<ECSAddress>, Exception> ecsCallback);

## Parameters

* address – an object of class ECSAddress which needs to be added
* ecsCallback – instance of ECSCallback containing list of ECSAddress and Exception

## Output

* Success response – ECSAddress object which contains list of addresses
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.createAndFetchAddress(address, **new** ECSCallback<List<ECSAddress>, Exception>() {  
 @Override  
 **public void** onResponse(<List<ECSAddress> result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSCountryCodeNotGiven (Please provide valid country code)
* ECSRegionisocode (Region selected is invalid) –
* ECSPostalCode (ZIP code selected is invalid)
* ECSfirstName **(**Please notice that the following characters are not allowed: \") ( \* ^ % $ # @ ! + , . ? / : ; { } [ ] ` ~ =)
* ECSlastName (Please notice that the following characters are not allowed: \") ( \* ^ % $ # @ ! + , . ? / : ; { } [ ] ` ~ =)
* ECSphone1 **(**Phone number selected is invalid)
* ECSaddressId **(**Given address doesn't exist or belong to another user)
* ECSValidationError **(**Data validation failed)

# **Fetch Saved Address**

This method is used to fetch the list of all saved addresses for the user.

## ECSServiceProvider

**void** fetchSavedAddresses(ECSCallback<List<ECSAddress>, Exception> ecsCallback);

## Parameters

* ecsCallback – instance of ECSCallback containing list of ECSAddress and Exception

## Output

* Success response – list of ECSAddress object which contains list of saved addresses
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.fetchSavedAddresses(**new** ECSCallback<List<ECSAddress>, Exception>() {  
 @Override  
 **public void** onResponse(List<ECSAddress> result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSNoSuchElementError (An error occurred while fetching the data from the server). Please re-login) – This error occurs when failed to get the response

# **Set Delivery Address**

This method is used to set the delivery address of the user.

## ECSServiceProvider

**void** setDeliveryAddress(ECSAddress address, ECSCallback<Boolean, Exception> ecsCallback);

## Parameters

* address – an object of class ECSAddress which has to be set as delivery address
* ecsCallback – instance of ECSCallback of type Boolean and Exception

## Output

* Success response – returns true if selected address is set as delivery address successfully
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices. setDeliveryAddress(addresses, **new** ECSCallback<Boolean, Exception>() {  
 @Override  
 **public void** onResponse(Boolean result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSUnsupportedDeliveryModeError ( Selected delivery mode is not supported)
* ECSaddressId **(**Given address doesn't exist or belong to another user)

# **Set and Fetch Delivery Address**

This method is used to set the delivery address of the user and fetch the address.

## ECSServiceProvider

**void** setAndFetchDeliveryAddress(ECSAddress address, ECSCallback<List<ECSAddress>, Exception> ecsCallback);

## Parameters

* address – an object of class ECSAddress which has to be set as delivery address
* ecsCallback – instance of ECSCallback containing list of ECSAddress and Exception

## Output

* Success response – returns the list of addresses
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.setAndFetchDeliveryAddress(address,**new** ECSCallback<List<ECSAddress>, Exception>() {  
 @Override  
 **public void** onResponse(List<ECSAddress> result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSUnsupportedDeliveryModeError ( Selected delivery mode is not supported)
* ECSaddressId **(**Given address doesn't exist or belong to another user)

# **Update Address**

This method is used to update the address of the user. This is used whenever the user edits the address.

## ECSServiceProvider

**void** updateAddress(**boolean** isDefaultAddress, ECSAddress address, ECSCallback<Boolean, Exception> ecsCallback);

## Parameters

* isDefaultAddress – Boolean value, if it is true the updated address is set as default address
* address – ECSAddress object for which the address is to be updated
* ecsCallback – instance of ECSCallback of type Boolean and Exception

## Output

* Success response – returns true if the address is updated successfully
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.updateAddress(**false**, addresses, **new** ECSCallback<Boolean, Exception>() {

@Override  
 **public void** onResponse(Boolean result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSUnsupportedDeliveryModeError ( Selected delivery mode is not supported)
* ECSCountryCodeNotGiven (Please provide valid country code)
* ECSRegionisocode (Region selected is invalid) –
* ECSPostalCode (ZIP code selected is invalid)
* ECSfirstName **(**Please notice that the following characters are not allowed: \") ( \* ^ % $ # @ ! + , . ? / : ; { } [ ] ` ~ =)
* ECSlastName (Please notice that the following characters are not allowed: \") ( \* ^ % $ # @ ! + , . ? / : ; { } [ ] ` ~ =)
* ECSphone1 **(**Phone number selected is invalid)
* ECSaddressId **(**Given address doesn't exist or belong to another user)
* ECSValidationError **(**Data validation failed)

# **Update and Fetch Address**

This method is used to update the address of the user and fetch the list of addresses.

## ECSServiceProvider

**void** updateAndFetchAddress(**boolean** isDefaultAddress, ECSAddress address, ECSCallback<List<ECSAddress>, Exception> ecsCallback);

## Parameters

* isDefaultAddress – Boolean value, if it is true the updated address is set as default address
* address – ECSAddress object for which the address is to be updated
* ecsCallback – instance of ECSCallback containing list of ECSAddress and Exception

## Output

* Success response – returns the list of addresses
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.updateAndFetchAddress(true, address**,new** ECSCallback<List<ECSAddress>, Exception>() {  
 @Override  
 **public void** onResponse(List<ECSAddress> result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSUnsupportedDeliveryModeError ( Selected delivery mode is not supported)
* ECSCountryCodeNotGiven (Please provide valid country code)
* ECSRegionisocode (Region selected is invalid) –
* ECSPostalCode (ZIP code selected is invalid)
* ECSfirstName **(**Please notice that the following characters are not allowed: \") ( \* ^ % $ # @ ! + , . ? / : ; { } [ ] ` ~ =)
* ECSlastName (Please notice that the following characters are not allowed: \") ( \* ^ % $ # @ ! + , . ? / : ; { } [ ] ` ~ =)
* ECSphone1 **(**Phone number selected is invalid)
* ECSaddressId **(**Given address doesn't exist or belong to another user)
* ECSValidationError **(**Data validation failed)

# **Delete Address**

This method is used to delete the address of the user.

## ECSServiceProvider

**void** deleteAddress(ECSAddress address, ECSCallback<Boolean, Exception> ecsCallback);

## Parameters

* address – ECSAddress object for which the address is to be deleted
* ecsCallback – instance of ECSCallback of type Boolean and Exception

## Output

* Success response – returns true if the address is deleted successfully
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.deleteAddress(address,**new** ECSCallback<Boolean, Exception>() {  
 @Override  
 **public void** onResponse(Boolean result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSaddressId **(**Given address doesn't exist or belong to another user)

# **Delete and Fetch Address**

This method is used to delete the address of the user and fetch the list of addresses.

## ECSServiceProvider

**void** deleteAndFetchAddress(@NonNull ECSAddress address,@NonNull ECSCallback<List<ECSAddress>, Exception> ecsCallback);

## Parameters

* address – ECSAddress object for which the address is to be deleted
* ecsCallback – instance of ECSCallback containing list of ECSAddress and Exception

## Output

* Success response – returns list of addresses
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.deleteAndFetchAddress(address, **new** ECSCallback<List<ECSAddress>, Exception>() {

@Override  
 **public void** onResponse(List<ECSAddress> result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSaddressId **(**Given address doesn't exist or belong to another user)

# **Fetch Retailers**

This method is used to fetch the list of available retailers for the selected product.

## ECSServiceProvider

**void** fetchRetailers(String ctn, ECSCallback<ECSRetailerList,Exception>ecsCallback);

## Parameters

* ctn – ctn of the selected product
* ecsCallback – instance of ECSCallback of type ECSRetailerList and Exception

## Output

* Success response – list of ECSRetailerList object which contains list of retailers
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices. fetchRetailers(ctn, **new** ECSCallback<ECSRetailerList, Exception>()  
 @Override  
 **public void** onResponse(ECSRetailerList result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSCtnNotProvided (Please provide the CTN) – occurs when ctn or not provided or invalid ctn is entered

Server Errors

* ECSCtnNotProvided (Please provide the CTN) – occurs when ctn is not provided or invalid ctn is entered

# **Fetch Retailers**

This method is used to fetch the list of available retailers for the selected product.

## ECSServiceProvider

**void** fetchRetailers(ECSProduct product, ECSCallback<ECSRetailerList,Exception> ecsCallback);

## Parameters

* product – ECSProduct object
* ecsCallback – instance of ECSCallback of type ECSRetailerList and Exception

## Output

* Success response – list of ECSRetailerList object which contains list of retailers
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.fetchRetailers(product, **new** ECSCallback<ECSRetailerList, Exception>(){  
 @Override  
 **public void** onResponse(ECSRetailerList result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSCtnNotProvided (Please provide the CTN) – occurs when ctn or not provided or invalid ctn is entered

Server Errors

* ECSCtnNotProvided (Please provide the CTN) – occurs when ctn is not provided or invalid ctn is entered

# **Set Payment Details**

This method is used to set payment details for the purchase.

## ECSServiceProvider

**void** setPaymentDetails(String paymentDetailsId, ECSCallback<Boolean, Exception> ecsCallback);

## Parameters

* paymentDetailsId – a string id representing the payment method
* ecsCallback – instance of ECSCallback of type Boolean and Exception

## Output

* Success response – returns true, if payment details is set successfully
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.setPaymentDetails(paymentId, **new** ECSCallback<Boolean, Exception>(){  
 @Override  
 **public void** onResponse(Boolean result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale
* ECSInvalidPaymentInfoError **(**Please provide valid payment details)

Server Errors

* ECSInvalidPaymentInfoError **(**Please provide valid payment details)

# **Fetch Payment Details**

This method is used to fetch list of existing payment methods.

## ECSServiceProvider

**void** fetchPaymentsDetails(ECSCallback<List<ECSPayment>, Exception> ecsCallback);

## Parameters

* ecsCallback – instance of ECSCallback containing list of ECSPayment and Exception

## Output

* Success response – ECSPayment object
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.fetchPaymentsDetails(**new** ECSCallback<List<ECSPayment>, Exception>() {  
 @Override  
 **public void** onResponse(List<ECSPayment> result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

# **Submit Order**

This method is used to place the order by entering cvv, after selecting the payment method.

## ECSServiceProvider

**void** submitOrder(String cvv, ECSCallback<ECSOrderDetail, Exception> ecsCallback);

## Parameters

* cvv – string value to be entered for placing the order
* ecsCallback – instance of ECSCallback of type ECSOrderDetail and Exception

## Output

* Success response – ECSOrderDetail object which contains details of order placed
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.submitOrder(pSecurityCode, **new** ECSCallback<ECSOrderDetail, Exception>() {  
 @Override  
 **public void** onResponse(ECSOrderDetail result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Server Errors

* ECSSessionCart (Please select your delivery address, delivery mode and payment method before submitting an order)

# **Make Payment**

This method is used to make the payment after entering all the payment details.

## ECSServiceProvider

**void** makePayment(ECSOrderDetail orderDetail, ECSAddress billingAddress, ECSCallback<ECSPaymentProvider, Exception> ecsCallback);

## Parameters

* orderDetail – object of class ECSOrderDetail which contains placed order details
* billingAddress – object of class ECSAddress which contains billing address
* ecsCallback – instance of ECSCallback of type ECSPaymentProvider and Exception

## Output

* Success response – ECSPaymentProvider object which contains world pay url
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.makePayment(orderDetail, billingAddress, **new** ECSCallback<ECSPaymentProvider, Exception>() {  
 @Override  
 **public void** onResponse(ECSPaymentProvider result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale
* ECSorderIdNil **(**Please provide valid order) – occurs for invalid order id

Server Errors

* ECSorderIdNil **(**Please provide valid order) – occurs for invalid order id
* ECSPaymentAuthorizationError (Payment authorization was not successful)

# **Fetch Order History**

This method is used to fetch the order history of the submitted orders.

## ECSServiceProvider

**void** fetchOrderHistory(**int** pageNumber, **int** pageSize, ECSCallback<ECSOrderHistory, Exception> ecsCallback);

## Parameters

* pageNumber – refer to the page number containing data
* pageSize – refer to the total number of pages
* ecsCallback – instance of ECSCallback of type ECSOrderHistory and Exception

## Output

* Success response – ECSOrderHistory object which contains details of placed order
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.fetchOrderHistory(pageNo, 0, **new** ECSCallback<ECSOrderHistory, Exception>() {  
 @Override  
 **public void** onResponse(ECSOrderHistory result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

# **Fetch Order Detail**

This method is used to fetch the details of given orders.

## ECSServiceProvider

**void** fetchOrderDetail(String orderId, ECSCallback<ECSOrderDetail,Exception> ecsCallback);

## Parameters

* orderId – string value which represents unique id for each order
* ecsCallback – instance of ECSCallback of type ECSOrderDetail and Exception

## Output

* Success response – ECSOrderDetail object which contains details of given order
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.fetchOrderDetail(orderNumber, **new** ECSCallback<ECSOrderDetail, Exception>() {  
 @Override  
 **public void** onResponse(ECSOrderDetail result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale
* ECSorderIdNil **(**Please provide valid order) – occurs for invalid order id

Server Errors

* ECSorderIdNil **(**Please provide valid order) – occurs for invalid order id

# **Fetch Order Detail**

This method is used to fetch the details of given orders.

## ECSServiceProvider

**void** fetchOrderDetail(ECSOrderDetail orderDetail,ECSCallback<ECSOrderDetail,Exception> ecsCallback);

## Parameters

* orderDetail – object of class ECSOrderDetail
* ecsCallback – instance of ECSCallback of type ECSOrderDetail and Exception

## Output

* Success response – ECSOrderDetail object which contains details of given order
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.fetchOrderDetail(orderDetail, **new** ECSCallback<ECSOrderDetail, Exception>() {  
 @Override  
 **public void** onResponse(ECSOrderDetail result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale
* ECSorderIdNil **(**Please provide valid order) – occurs for invalid order id

Server Errors

* ECSorderIdNil **(**Please provide valid order) – occurs for invalid order id

# **Fetch Order Detail**

This method is used to fetch the details of given orders.

## ECSServiceProvider

**void** fetchOrderDetail(ECSOrders orders, ECSCallback<ECSOrders, Exception> ecsCallback);

## Parameters

* orders – object of class ECSOrders
* ecsCallback – instance of ECSCallback of type ECSOrders and Exception

## Output

* Success response – ECSOrders object which contains details of given order
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.fetchOrderDetail(orders, **new** ECSCallback<ECSOrders, Exception>() {  
 @Override  
 **public void** onResponse(ECSOrders result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale
* ECSorderIdNil **(**Please provide valid order) – occurs for invalid order id

Server Errors

* ECSorderIdNil **(**Please provide valid order) – occurs for invalid order id

# **Fetch User Profile**

This method is used to fetch the user details.

## ECSServiceProvider

**void** fetchUserProfile(ECSCallback<ECSUserProfile,Exception> ecsCallback);

## Parameters

* ecsCallback – instance of ECSCallback of type ECSUserProfile and Exception

## Output

* Success response – ECSUserProfile object which contains details of user
* Failure response – Exception (contains localized error message) and ECSError which contains error code and error type.

## Sample

**appInfra** = **new** AppInfra.Builder().build(getApplicationContext());

ECSServices ecsServices = **new** ECSServices( **"your propositionID"**, appInfra);  
  
ecsServices.fetchUserProfile(**new** ECSCallback<ECSUserProfile, Exception>() {  
 @Override  
 **public void** onResponse(ECSUserProfile result) {

}  
  
 @Override  
 **public void** onFailure(Exception error, ECSError ecsError) {  
  
 }  
});

## Errors

Client Errors

* ECSLocaleNotFound (Please set the locale) – call configureECS or configureECSToGetConfiguration to set locale
* ECSSiteIdNotFound (Philips shop is not available for the selected country, only retailer mode is available) – occurs when failed to get config or Hybris not present
* ECSBaseURLNotFound (Base URL not found) - occurs when Hybris is not present for the given locale

Common Errors

* ECSinvalid\_grant (We have encountered technical glitch (Invalid Janrain Token). Please re-login) – This error occurs when the Janrain token is invalid
* ECSinvalid\_client (We have encountered technical glitch (Bad client credentials). Please re-login)
* ECSUnsupported\_grant\_type (We have encountered technical glitch (Unsupported grant type). Please re-login)
* ECSOAuthNotCalled (User authorization is required for this action)
* Whenever these error are encountered call hybrisRefreshOAuth or re-login.