MobileSDK Reference

Programming Guides

Class References

Constant References

Category References

Changelog Document

- **SDK 2.0**
- Support decryption of remote notification
- Changed the way the sdk is initialized from inside a notification extension
- **SDK 1.9**
- Support action button in push notification center
- **SDK 1.8**
- Support push only without in app message
- Support deeplinking and main action when user click on push notification
- **SDK 1.7.4**
- Improve how In App Content are managed in cache
- Correction on dismiss of inapp content viewcontroller when it is not wrapped inside other controller and when a link is present
- **SDK 1.7.3**
- Improve behavior and robustness of the way In App messages are fetch
- Add a type for In App refresh only for dev use: .Minutely
- **SDK 1.7.2**
- Correction on IAC cache issue when app was killed (file was not updated)
- **SDK 1.7.1**
- Correction on duplicate symbol due to integrate library
- **SDK 1.7**
- Added geolocation functionality
- Misc Bugs correction
- Add http header to inform which version of the platform is supported
- Support of bitcode

- **SDK 1.6**
  - iOS 11 support
  - misc bug corrections
  - consolidate received event
  - adapt user-agent of request
- **SDK 1.5.2**
  - correct bug for in app content that must be displayed only once
- **SDK 1.5.1**
  - correct crash bug that happens when expiration or creation date for in app content is null
- **SDK 1.5**
  - sendDeviceInfo deprecated method replaced with sendDeviceInfo:(SMDeviceInfos*)deviceInfos method
  - New SMManager category for DataTransaction with back-end
  - public SMDeviceInfos object
  - iOS 10 support of UserNotifications framework
  - stop supporting of iOS 7
  - cache on last sent UserCustomEvent
  - Update deviceID with the one received from platform
- **SDK 1.4.5**
  - Store last sent user custom event and check if a modification has been done before sending next one
- **SDK 1.4.4**
  - compare device token based on string instead of NSData (bug swift 3 and Data class)
- **SDK 1.4.3**
  - correction for max number of InApp Content crash when max > number of messages received
  - creation date of in app content
  - dismiss when no button in notification payload
- **SDK 1.4.2**
  - correction on unregisterForRemoteNotification method
  - issue with the notification when the application was killed
- **SDK 1.4.1**
  - bug corrections
- **SDK 1.4**
  - enum SMInAppMessageRefreshType has been renamed in SMInAppRefreshType (this enum is used both for InApp Message and for InApp Content) possible values are :
    - kSMIA_RefreshType_None

        - kSMIA_RefreshType_Hourly
        - kSMIA_RefreshType_Daily

- **SDK 1.3**

    - To access easily all API methods you will need to replace #import SMManager.h by #import SMHelper.h

- **SDK 1.2**

    - The API sendEvent: has been renamed to sendSMEvent: (This call will prevent compilation)
    - The API registerForRemoteNotification has been added. It allows applications to register remote-notification when they really need it. This, then, becomes a mandatory call for receiving pushes (after starting the library).

SDK programming guide Document

The following *Mandatory* section must be implemented in order to succesfully start the SDK and start receiving remote push notificaitons from your back-end.

Mandatory :

SMManager is the main entry point and will inform you about how to start the library.
SMManager(RemoteNotification) are the APIs wich will keep your application in sync with the library.

Optional :

SMManager(DataTransaction) explains how you can send or retrieve specific and predefined information to the back-end.
SMManager(UserNotification) are the APIs wich will allow to be fully compatible with iOS 10.
SMManager(SMEvent) explains how the application can communicate with the back-end.
SMManager(SilentPush) will guide you in order to implement silent-push-notifications.
SMManager(InAppMessage) gives an explanation about the IAM-service and how to configure it.
SMManager(InAppContent) gives an explanation about the IAC-service and how to configure it.
SMManager(Log) explains how the library log messages. May be helpful for debugging.
SMManager(StyleOptions) will provide methods to set and reset In App Content style options.

SMNSNotification Document

Notifications :

- **kSMNotification_Event_ButtonClicked**

  NSString representing a notification name you can listen to.

  An NSNotification with this name is broadcasted when the user interacts with a remote-notification.

  Usefull to retrieve user's actions on a received remote-notification, developers may listen to kSMNotification_Event_ButtonClicked from NSNotificationCenter.

- **kSMNotification_Event_WillDisplayNotification**

  NSString representing a notification name you can listen to.

  An NSNotification with this name is broadcasted shortly before displaying a remote-notification.

  Primary-application may use this notification to pause any ongoing work before the remote-notification is displayed.

  This notification-name is also triggered even if you disable shouldDisplayRemoteNotification (see SMManagerSetting).

- **kSMNotification_Event_WillDismissNotification**

  NSString representing a notification name you can listen to.

  An NSNotification with this name is broadcasted shortly before Dismissing the current remote-notification.

  Primary-application may use this notification to resume any paused work. (see kSMNotification_Event_WillDisplayNotification).

- **kSMNotification_Event_DidReceiveRemoteNotification**

  NSString representing a notification name you can listen to.

  An NSNotification with this name is broadcasted shortly after receiving a remote-notification.

Primary-application may use this notification to decide when to display any remote-notification.

- **kSMNotification_Event_DidReceiveInAppMessage**

  NSString representing a notification name you can listen to. An NSNotification with this name is broadcasted shortly after receiving InApp messages.

  Primary-application may use this notification to manage the received InApp messages.

- **kSMNotification_Event_DidReceiveInAppContent**

  NSString representing a notification name you can listen to.

  An NSNotification with this name is broadcasted shortly after receiving InApp contents.

  Primary-application may use this notification to manage the received InApp contents.

Data :

- **kSMNotification_Data_ButtonData**

  NSString representing a key to retrieve an object inside NSNotification.

  Use the key kSMNotification_Data_ButtonData to retrieve the object SMNotificationButtonData from the NSNotification-name kSMNotification_Event_ButtonClicked.

- **kSMNotification_Data_RemoteNotification**

  NSString representing a key to retrieve an object inside NSNotification.

  Use the key kSMNotification_Data_RemoteNotification to retrieve an NSDictionary instance with push ID and name

- **kSMNotification_Data_InAppMessage**

  NSString representing a key to retrieve an object inside NSNotification.

  Use the key kSMNotification_Data_InAppMessage to retrieve an NSDictionary instance with an array of SMNotificationMessage

kSMNotification_Data_InAppContent

- NSString representing a key to retrieve an object inside NSNotification.

  Use the key kSMNotification_Data_InAppContent to retrieve an NSDictionary instance with an array with informations about number of In App contents by category

iOS 9 special considerations Document

- Starting iOS 9, https is the default protocol to adopt for all network calls. Please check Apple-documentation for additional information.

  If you're not adopting https yet, warning messages will appear in your console. In order to avoid these, you may add the following keys in you info.plist:

  - NSAppTransportSecurity as a dictionary
  - NSAllowsArbitraryLoads as BOOL set to TRUE (inside dictionary NSAppTransportSecurity)

- iOS 9 and links management

  According to iOS 9 documentation, apple is more and more using universal standard HTTP or HTTPS links instead of URL schemes.

  To learn more about universal links and configure properly your app, see Support Universal Links for additional information.

  However you can still continue to use url scheme to open other third party apps, know that you will need to provide a specific whitelist in your app .plist file

  This white list will be like for example :

  OBJECTIVE-C

  ```
  <key>LSApplicationQueriesSchemes</key>
  <array>
      <string>fbapi</string>
      <string>fbauth2</string>
      <string>fbshareextension</string>
      <string>fb-messenger-api</string>
      <string>twitter</string>
      <string>skype</string>
      <string>whatsapp</string>
      <string>wechat</string>
      <string>line</string>
      <string>instagram</string>
      <string>kakaotalk</string>
      <string>comgooglemaps</string>
  </array>
  ```

SMBaseMessage Class Reference

Inherits from	NSObject
Declared in	SMBaseMessage.h

Overview

This is the Base class for message (push notif, in app message and in app content)

SMBaseMessage :

idMessage

NSString instance providing the id of the message

OBJECTIVE-C
```
@property (nonatomic) NSString *idMessage
```

Declared In
SMBaseMessage.h

dateCreation

NSDate instance providing the creation date of the message

OBJECTIVE-C
```
@property (nonatomic) NSDate *dateCreation
```

Declared In
SMBaseMessage.h

SMDeviceInfos Class Reference

Inherits from	NSObject
Declared in	SMDeviceInfos.h

Overview

SMDeviceInfos :

`externalId`

Any External Id in your possession that you wish to transmit to the back-end

OBJECTIVE-C

```
@property (nonatomic) NSString *externalId
```

Discussion
NSString By default, it is an empty string

Declared In
`SMDeviceInfos.h`

`+ defaultDeviceInfos`

Create a generic deviceinfos object

OBJECTIVE-C

```
+ (instancetype)defaultDeviceInfos
```

Return Value
a SMDeviceInfos object

Declared In

`SMDeviceInfos.h`

+ deviceInfosWithExternalId:

Create a deviceinfos object with an external id property already set with the value provided

OBJECTIVE-C

```
+ (instancetype)deviceInfosWithExternalId:(NSString *)externalId
```

Parameters

externalId	a string containing the id you want to provide to the back-end

Return Value

a SMDeviceInfos object

Declared In

`SMDeviceInfos.h`

SMEvent Class Reference

Inherits from	NSObject
Declared in	SMEvent.h

Overview

This is the Base class for all type of events

SMEvent :

`shouldCache`

Confirm if the current event should be cached or not

OBJECTIVE-C

```
@property (nonatomic) BOOL shouldCache
```

Discussion
If the event fail to be delivered to your backend, then by default, it is cached into an internal queue. After a while, the library will automaticly try to send it again. Should you want to prevent this behaviour, feel free to set this property to FALSE. By default, it is set to TRUE

Declared In
`SMEvent.h`

`+ eventWithDictionary:`

Create a generic event object that will be sent to platform

OBJECTIVE-C

```
+ (instancetype)eventWithDictionary:(NSDictionary *)dict
```

Parameters

dict	a Dictionary containing any kind of custom datas that must be stored and managed by platform

Return Value
a SMEvent object

Declared In
`SMEvent.h`

– applyBlockSuccess:BlockFailure:

Allow to initialise a success block and/or a failure block that will be triggered after an event is sent to the platform

OBJECTIVE-C
```
- (void)applyBlockSuccess:(SMCompletionBlockSuccess)blockSuccess BlockFailure:
(SMCompletionBlockFailure)blockFailure
```

Parameters

blockSuccess	a SMCompletionBlockSuccess block that will be triggered if the send to the platform is successfull
blockFailure	a SMCompletionBlockFailure block that will be triggered if the send to the platform has failed

Discussion
This method may be used as follow:

@code NSDictionary *dictMyCustomData = @{@“MyKey”: @“MyValue”}; SMEvent* event = [SMEvent eventWithDictionary:dictMyCustomData]; [event applyBlockSuccess:^(SMSuccess *success) { // My code for success} BlockFailure:^(SMFailure* failure) { // My code for failure }]; [[SMManager sharedInstance] sendSMEvent:event]; @endcode This method is optional. Use it only if you need it.

Warning: It is developer’s responsability to make sure no strong retrain cycles are added to the completion-blocks. Make sure to read the following Apple documentation.

Declared In
`SMEvent.h`

SMEventUser Class Reference

Inherits from	SMEvent : NSObject
Declared in	SMEventUser.h

Overview

@inherits SMEvent

Class representing all User-eventing Instances of this class should not be directly created. Please use children classes instead.

SMEventUser:

SMEventUserLogin Class Reference

Inherits from	SMEventUser : SMEvent : NSObject
Declared in	SMEventUserLogin.h

Overview

@class SMEventUserLogin @inherits SMEventUser

user login event class

SMEventUserLogin:

+ eventWithEmail:

Create a UserLogin event object that will be sent to selligent platform when user logged in

OBJECTIVE-C

```
+ (instancetype)eventWithEmail:(NSString *)mail
```

Parameters

mail	the e-mail of the user

Return Value
a SMEventUserLogin object

Declared In
`SMEventUserLogin.h`

+ eventWithEmail:Dictionary:

Create a UserLogin event object that will be sent to selligent platform when user logged in

OBJECTIVE-C

```
+ (instancetype)eventWithEmail:(NSString *)mail Dictionary:(NSDictionary<NSString*,NSString*>
*)dict
```

Parameters

mail	the e-mail of the user
dict	a Dictionary containing a string as data that must be stored and managed by platform If email is not provided you can use in the dictionary an alternate key/value field to search for the user example: [SMEventUserLogin eventWithEmail @“” Dictionary: @{@“userID”: @“1234”}];

Return Value

a SMEventUserLogin object

Declared In

`SMEventUserLogin.h`

SMEventUserLogout Class Reference

Inherits from	SMEventUser : SMEvent : NSObject
Declared in	SMEventUserLogout.h

Overview

@class SMEventUserLogout @inherits SMEventUser

user logout event class

SMEventUserLogout:

`+ eventWithEmail:`

Create a UserLogout event object that will be sent to selligent platform when user logged out

OBJECTIVE-C

```
+ (instancetype)eventWithEmail:(NSString *)mail
```

Parameters

mail	the e-mail of the user

Return Value
a SMEventUserLogout object

Declared In
`SMEventUserLogout.h`

`+ eventWithEmail:Dictionary:`

Create a UserLogout event object that will be sent to selligent platform when user logged out

OBJECTIVE-C

```
+ (instancetype)eventWithEmail:(NSString *)mail Dictionary:(NSDictionary<NSString*,NSString*>
*)dict
```

Parameters

mail	the e-mail of the user
dict	a Dictionary containing a string as data that must be stored and managed by platform If email is not provided you can use in the dictionary an alternate key/value field to search for the user example: [SMEventUserLogout eventWithEmail @“” Dictionary: @{@“userID”: @“1234”}];

Return Value

a SMEventUserLogout object

Declared In

`SMEventUserLogout.h`

SMEventUserRegistration Class Reference

Inherits from	SMEventUser : SMEvent : NSObject
Declared in	SMEventUserRegistration.h

Overview

@class SMEventUserRegistration @inherits SMEventUser

user registration event class

SMEventUserRegistration:

+ eventWithEmail:

Create a UserRegistration event object that will be sent to selligent platform when user registered

OBJECTIVE-C

```
+ (instancetype)eventWithEmail:(NSString *)mail
```

Parameters

mail	the e-mail of the user

Return Value
a SMEventUserRegistration object

Declared In
SMEventUserRegistration.h

+ eventWithEmail:Dictionary:

Create a UserRegistration event object that will be sent to selligent platform when user registered

OBJECTIVE-C

```
+ (instancetype)eventWithEmail:(NSString *)mail Dictionary:(NSDictionary<NSString*,NSString*>
*)dict
```

Parameters

mail	the e-mail of the user
dict	a Dictionary containing a string a string as data that must be stored and managed by platform If email is not provided you can use in the dictionary an alternate key/value field to search for the user example: [SMEventUserRegistration eventWithEmail @“” Dictionary: @{@“userID”: @“1234”}];

Return Value

a SMEventUserRegistration object

Declared In

`SMEventUserRegistration.h`

SMEventUserUnregistration Class Reference

Inherits from	SMEventUser : SMEvent : NSObject
Declared in	SMEventUserUnregistration.h

Overview

@class SMEventUserUnregistration @inherits SMEventUser

user unregistration event class

SMEventUserUnregistration:

+ eventWithEmail:

Create a UserUnregistration event object that will be sent to selligent platform when user unregistered

OBJECTIVE-C

```
+ (instancetype)eventWithEmail:(NSString *)mail
```

Parameters

mail	the e-mail of the user

Return Value
a SMEventUserUnregistration object

Declared In
`SMEventUserUnregistration.h`

+ eventWithEmail:Dictionary:

Create a UserUnregistration event object that will be sent to selligent platform when user unregistered

OBJECTIVE-C

```
+ (instancetype)eventWithEmail:(NSString *)mail Dictionary:(NSDictionary<NSString*,NSString*>
*)dict
```

Parameters

mail	the e-mail of the user
dict	a Dictionary containing a String that must be stored and managed by platform

Return Value

a SMEventUserUnregistration object

Declared In

`SMEventUserUnregistration.h`

SMFailure Class Reference

Inherits from	SMMessage : NSObject
Declared in	SMFailure.h

Overview

This class is used to return any error

SMFailure:

SMInAppContentHTMLViewController Class Reference

Inherits from	SMInAppContentViewController : UIViewController
Declared in	SMInAppContentHTMLViewController.h

Overview

a view controller for HTML In App Content

SMInAppContentHTMLViewController :

+ viewControllerForCategory:

This will provide you a custom viewcontroller with all HTML in app content for a specific category

OBJECTIVE-C

```
+ (instancetype)viewControllerForCategory:(NSString *)category
```

Parameters

category	a NSString of the desired category of In App Content

Return Value

a SMInAppContentHTMLViewController

Discussion

The viewcontroller will take all available space in screen and will contain a close button if it is presented as it is. Otherwise the app will have to define a UIContainerView which will contain the view controller

Declared In

`SMInAppContentHTMLViewController.h`

+ viewControllerForCategory:AndOptions:

This will provide you a custom viewcontroller with all HTML in app content for a specific category

OBJECTIVE-C

```
+ (instancetype)viewControllerForCategory:(NSString *)category AndOptions:
(SMInAppContentStyleOptions *)options
```

Parameters

category	a NSString of the desired category of In App Content
options	a SMInAppContentStyleOptions object allowing you to customise the in app content

Return Value
a SMInAppContentHTMLViewController

Discussion
The viewcontroller will take all available space in screen and will contain a close button if it is presented as it is. Otherwise the app will have to define a UIContainerView which will contain the view controller

Declared In
`SMInAppContentHTMLViewController.h`

+ viewControllerForCategory:InNumberOfBoxes:

This will provide you a custom viewcontroller with HTML in app content for a specific category

OBJECTIVE-C

```
+ (instancetype)viewControllerForCategory:(NSString *)category InNumberOfBoxes:(int)numberofboxes
```

Parameters

category	a NSString of the desired category of In App Content
numberofboxes	an int corresponding to the maximum numbers of html boxes that the view controller must contain

Return Value
a SMInAppContentHTMLViewController

Discussion
The viewcontroller will take all available space in screen and will contain a close button if it is presented as it is. Otherwise the app will have to provide a UIContainerView which will contain the view controller

Declared In
`SMInAppContentHTMLViewController.h`

+ viewControllerForCategory:InNumberOfBoxes:AndOptions:

This will provide you a custom viewcontroller with HTML in app content for a specific category

OBJECTIVE-C

```
+ (instancetype)viewControllerForCategory:(NSString *)category InNumberOfBoxes:(int)numberofboxes
AndOptions:(SMInAppContentStyleOptions *)options
```

Parameters

category	a NSString of the desired category of In App Content
numberofboxes	an int corresponding to the maximum numbers of html boxes that the view controller must contain
options	a SMInAppContentStyleOptions object allowing you to customise the in app content

Return Value
a SMInAppContentHTMLViewController

Discussion
The viewcontroller will take all available space in screen and will contain a close button if it is presented as it is. Otherwise the app will have to provide a UIContainerView which will contain the view controller

Declared In
`SMInAppContentHTMLViewController.h`

SMInAppContentImageViewController Class Reference

Inherits from	SMInAppContentViewController : UIViewController
Declared in	SMInAppContentImageViewController.h

Overview

a view controller for Image In App Content

SMInAppContentImageViewController :

`+ viewControllerForCategory:`

This will provide you a custom viewcontroller with one image view loaded with the url provided by an in app content for a specific category of image type

OBJECTIVE-C

```
+ (instancetype)viewControllerForCategory:(NSString *)category
```

Parameters

category	a NSString of the desired category of In App Content

Return Value

a SMInAppContentImageViewController

Discussion

The viewcontroller will take all available space in screen and will contain a close button if it is presented as it is. Otherwise the app will have to provide a UIContainerView which will contain the view controller

Declared In

`SMInAppContentImageViewController.h`

`+ viewControllerForCategory:AndOptions:`

This will provide you a custom viewcontroller with one web view loaded with the url provided by an in app content for a specific category of an Image type

OBJECTIVE-C

```
+ (instancetype)viewControllerForCategory:(NSString *)category AndOptions:
(SMInAppContentStyleOptions *)options
```

Parameters

category	a NSString of the desired category of In App Content
options	a SMInAppContentStyleOptions object allowing you to customise the in app content

Return Value

a SMInAppContentImageViewController

Discussion

The viewcontroller will take all available space in screen and will contain a close button if it is presented as it is. Otherwise the app will have to define a UIContainerView which will contain the view controller

Declared In

`SMInAppContentImageViewController.h`

SMInAppContentMessage Class Reference

Inherits from	SMBaseMessage : NSObject
Declared in	SMInAppContentMessage.h

Overview

SMInAppContentMessage :

`title`

NSString instance providing the title of the message

OBJECTIVE-C

```
@property (nonatomic) NSString *title
```

Declared In

`SMInAppContentMessage.h`

`body`

NSString instance providing the content of the message

OBJECTIVE-C

```
@property (nonatomic) NSString *body
```

Declared In

`SMInAppContentMessage.h`

`category`

NSString instance providing the category of the message

OBJECTIVE-C

```
@property (nonatomic) NSString *category
```

Declared In
SMInAppContentMessage.h

iacType

SMInAppContentType instance providing the in app content type of the message

OBJECTIVE-C

```
@property (nonatomic) SMInAppContentType iacType
```

Declared In
SMInAppContentMessage.h

contentExpiration

NSDate instance providing the expiration date of the message

OBJECTIVE-C

```
@property (nonatomic) NSDate *contentExpiration
```

Declared In
SMInAppContentMessage.h

arrayIACLinks

NSArray of SMLink objects

OBJECTIVE-C

```
@property (nonatomic) NSArray *arrayIACLinks
```

Declared In
SMInAppContentMessage.h

SMInAppContentStyleOptions Class Reference

Inherits from	NSObject
Declared in	SMInAppContentStyleOptions.h

Overview

SMInAppContentStyleOptions :

mainViewIsScrollable

inform the sdk if the main container view of your in app content must be scrollable

OBJECTIVE-C

```
@property (nonatomic) bool mainViewIsScrollable
```

Discussion
bool By default, it is set to true

Declared In
`SMInAppContentStyleOptions.h`

mainViewBackgroundColor

Set the main container view of your in app contents background color

OBJECTIVE-C

```
@property (nonatomic) UIColor *mainViewBackgroundColor
```

Discussion
UIColor By default, it is clearColor

Declared In

`SMInAppContentStyleOptions.h`

`activityIndicatorStyle`

Set the UIActivityIndicator style

OBJECTIVE-C
```
@property (nonatomic) UIActivityIndicatorViewStyle activityIndicatorStyle
```

Discussion
UIActivityIndicatorViewStyle By default, it is UIActivityIndicatorViewStyleGray

Declared In
`SMInAppContentStyleOptions.h`

`isStatusBarHidden`

Set the boolean to determine if status bar must be hidden or not

OBJECTIVE-C
```
@property (nonatomic) bool isStatusBarHidden
```

Discussion
bool By default, it is NO

Declared In
`SMInAppContentStyleOptions.h`

`boxLeading`

Set the leading constant between edge of view and every in app content box

OBJECTIVE-C
```
@property (nonatomic) CGFloat boxLeading
```

Discussion
CGFloat must be a positive value By default, it is set to 10

Declared In
`SMInAppContentStyleOptions.h`

boxTrailing

Set the trailing constant between edge of view and every in app content box

OBJECTIVE-C
```
@property (nonatomic) CGFloat boxTrailing
```

Discussion
CGFloat must be a positive value By default, it is set to 10

Declared In
`SMInAppContentStyleOptions.h`

marginBetweenBoxes

Set the Margin between bottom of a box and top of next one

OBJECTIVE-C
```
@property (nonatomic) CGFloat marginBetweenBoxes
```

Discussion
CGFloat must be a positive value By default, it is set to 20

Declared In
`SMInAppContentStyleOptions.h`

marginBetweenFirstBoxAndTopOfView

Set the Margin between top of first box and top of view

OBJECTIVE-C
```
@property (nonatomic) CGFloat marginBetweenFirstBoxAndTopOfView
```

Discussion
CGFloat must be a positive value By default, it is set to 20

Declared In
`SMInAppContentStyleOptions.h`

marginBetweenLastBoxAndBottomOfView

Set the Margin between bottom of last box and bottom of view

OBJECTIVE-C

```
@property (nonatomic) CGFloat marginBetweenLastBoxAndBottomOfView
```

Discussion
CGFloat must be a positive value By default, it is set to 20

Declared In
`SMInAppContentStyleOptions.h`

boxBorderWidth

Set the border width for all boxes

OBJECTIVE-C

```
@property (nonatomic) CGFloat boxBorderWidth
```

Discussion
CGFloat must be a positive value By default, it is set to 1

Declared In
`SMInAppContentStyleOptions.h`

boxBorderColor

Set the color of box Border

OBJECTIVE-C

```
@property (nonatomic) UIColor *boxBorderColor
```

Discussion
UIColor By default, it is set to [UIColor colorWithRed:0.5 green:0.5 blue:0.5 alpha:0.8]

Declared In
`SMInAppContentStyleOptions.h`

boxCornerRadius

Set the radius of the corner for all boxes

OBJECTIVE-C

```
@property (nonatomic) CGFloat boxCornerRadius
```

Discussion
CGFloat must be a positive value By default, it is set not set

Declared In
`SMInAppContentStyleOptions.h`

`boxBackgroundColor`

Set the background color of all boxes

OBJECTIVE-C
```
@property (nonatomic) UIColor *boxBackgroundColor
```

Discussion
UIColor By default, it is clearColor

Declared In
`SMInAppContentStyleOptions.h`

`boxShadowColor`

Set the shadow color of all boxes

OBJECTIVE-C
```
@property (nonatomic) UIColor *boxShadowColor
```

Discussion
UIColor By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

`boxShadowOpacity`

Set the shadow opacity of all boxes

OBJECTIVE-C
```
@property (nonatomic) CGFloat boxShadowOpacity
```

Discussion

CGFloat By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

boxShadowRadius

Set the shadow radius of all boxes

OBJECTIVE-C
```
@property (nonatomic) CGFloat boxShadowRadius
```

Discussion
CGFloat By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

boxShadowOffset

Set the shadow offset of all boxes

OBJECTIVE-C
```
@property (nonatomic) CGSize boxShadowOffset
```

Discussion
CGSize By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

titleBorderWidth

Set the border width for title

OBJECTIVE-C
```
@property (nonatomic) CGFloat titleBorderWidth
```

Discussion
CGFloat By default, it is not set

Declared In

`SMInAppContentStyleOptions.h`

`titleBorderColor`

Set the color of title border

OBJECTIVE-C

```
@property (nonatomic) UIColor *titleBorderColor
```

Discussion
UIColor By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

`titleCornerRadius`

Set the radius of the corner for all boxes

OBJECTIVE-C

```
@property (nonatomic) CGFloat titleCornerRadius
```

Discussion
CGFloat By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

`titleBackgroundColor`

Set the background color of all titles

OBJECTIVE-C

```
@property (nonatomic) UIColor *titleBackgroundColor
```

Discussion
UIColor By default, it is whiteColor

Declared In
`SMInAppContentStyleOptions.h`

titleNumberOfLines

Set the number of lines of all titles

OBJECTIVE-C
```
@property (nonatomic) CGFloat titleNumberOfLines
```

Discussion
UIColor By default, it is 0

Declared In
`SMInAppContentStyleOptions.h`

titleLineBreakMode

Set the NSLineBreakMode of all titles

OBJECTIVE-C
```
@property (nonatomic) NSLineBreakMode titleLineBreakMode
```

Discussion
NSLineBreakMode By default, it is NSLineBreakByWordWrapping

Declared In
`SMInAppContentStyleOptions.h`

titleTextAlignment

Set the title text alignment

OBJECTIVE-C
```
@property (nonatomic) NSTextAlignment titleTextAlignment
```

Discussion
NSTextAlignment By default, it is NSTextAlignmentLeft

Declared In
`SMInAppContentStyleOptions.h`

titleAttributes

Set the attributes that will be passed to NSAttributedString init which will create the text that will be display for title

OBJECTIVE-C
```
@property (nonatomic) NSDictionary *titleAttributes
```

Discussion
NSDictionary By default, it is nil

Declared In
`SMInAppContentStyleOptions.h`

titleTextColor

Set title text color

OBJECTIVE-C
```
@property (nonatomic) UIColor *titleTextColor
```

Discussion
UIColor By default, it is iOS default

Declared In
`SMInAppContentStyleOptions.h`

titleFont

Set font of the title

OBJECTIVE-C
```
@property (nonatomic) UIFont *titleFont
```

Discussion
UIFont By default, it is iOS default

Declared In
`SMInAppContentStyleOptions.h`

titleTrailing

Set the trailing between the title container and the box

OBJECTIVE-C

```
@property (nonatomic) CGFloat titleTrailing
```

Discussion
CGFloat By default, it is 10.0

Declared In
`SMInAppContentStyleOptions.h`

`titleLeading`

Set the leading between the title container and the box

OBJECTIVE-C
```
@property (nonatomic) CGFloat titleLeading
```

Discussion
CGFloat By default, it is 10.0

Declared In
`SMInAppContentStyleOptions.h`

`titleTop`

Set the top between the title container and the box

OBJECTIVE-C
```
@property (nonatomic) CGFloat titleTop
```

Discussion
CGFloat By default, it is 30.0

Declared In
`SMInAppContentStyleOptions.h`

`titleShadowColor`

Set the shadow color of all titles

OBJECTIVE-C
```
@property (nonatomic) UIColor *titleShadowColor
```

Discussion

UIColor By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

titleShadowOpacity

Set the shadow opacity of all titles

OBJECTIVE-C
```
@property (nonatomic) CGFloat titleShadowOpacity
```

Discussion
CGFloat By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

titleShadowRadius

Set the corner radius of all titles

OBJECTIVE-C
```
@property (nonatomic) CGFloat titleShadowRadius
```

Discussion
CGFloat By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

titleShadowOffset

Set the shadow offset of all titles

OBJECTIVE-C
```
@property (nonatomic) CGSize titleShadowOffset
```

Discussion
CGSize By default, it is not set

Declared In

SMInAppContentStyleOptions.h

showTitleBorderBottom

Set the bool that will tell if a border bottom must be displayed under all titles in box

OBJECTIVE-C

```
@property (nonatomic) bool showTitleBorderBottom
```

Discussion
bool By default, it is NO

Declared In
SMInAppContentStyleOptions.h

titleBorderBottomColor

Set the border color of all border bottom that are displayed under all titles in box

OBJECTIVE-C

```
@property (nonatomic) UIColor *titleBorderBottomColor
```

Discussion
UIColor By default, it is not set

Declared In
SMInAppContentStyleOptions.h

textViewTrailing

Set the trailing between the textview and the box

OBJECTIVE-C

```
@property (nonatomic) CGFloat textViewTrailing
```

Discussion
CGFloat By default, it is 10.0

Declared In
SMInAppContentStyleOptions.h

textViewLeading

Set the leading between the textview and the box

OBJECTIVE-C

```
@property (nonatomic) CGFloat textViewLeading
```

Discussion
CGFloat By default, it is 10.0

Declared In
`SMInAppContentStyleOptions.h`

textViewTop

Set the top between the textview and the box

OBJECTIVE-C

```
@property (nonatomic) CGFloat textViewTop
```

Discussion
CGFloat By default, it is 10.0

Declared In
`SMInAppContentStyleOptions.h`

textViewContentOffset

Set the textview content offset

OBJECTIVE-C

```
@property (nonatomic) CGPoint textViewContentOffset
```

Discussion
CGPoint By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

textViewContentInset

Set the textview content edge inset

OBJECTIVE-C
```
@property (nonatomic) UIEdgeInsets textViewContentInset
```

Discussion
UIEdgeInsets By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

textViewBorderWidth

Set the border width for textview

OBJECTIVE-C
```
@property (nonatomic) CGFloat textViewBorderWidth
```

Discussion
CGFloat By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

textViewBorderColor

Set the color of textview Border

OBJECTIVE-C
```
@property (nonatomic) UIColor *textViewBorderColor
```

Discussion
UIColor By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

textViewCornerRadius

Set the radius of the corner for all textview

OBJECTIVE-C

```
@property (nonatomic) CGFloat textViewCornerRadius
```

Discussion
CGFloat By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

`textViewBackgroundColor`

Set the background color of textview

OBJECTIVE-C
```
@property (nonatomic) UIColor *textViewBackgroundColor
```

Discussion
UIColor By default, it is whiteColor

Declared In
`SMInAppContentStyleOptions.h`

`linksAlignment`

Set position of the links, this can be Left, Right, or Center

OBJECTIVE-C
```
@property (nonatomic) SMContentAlignment linksAlignment
```

Discussion
SMContentAlignment By default, it is kSMAlignLeft

Declared In
`SMInAppContentStyleOptions.h`

`linksMargin`

Set the constant margin between links and edge of box (depends also of the linksAlignment property: if linksAligment is kSMAlignLeft than this property will only be applied for Leading margin, if linksAlignment is kSMAlignRight than this property is applied to trailing margin, if linksAlignment is kSMAlignCenter than this property is applied both for leading and trailing)

OBJECTIVE-C

```
@property (nonatomic) CGFloat linksMargin
```

Discussion
CGFloat By default, it is 10

Declared In
`SMInAppContentStyleOptions.h`

`linksTop`

Set the constant between links top and bottom of textview

OBJECTIVE-C
```
@property (nonatomic) CGFloat linksTop
```

Discussion
CGFloat By default it is 10

Declared In
`SMInAppContentStyleOptions.h`

`linksBottom`

Set the constant between links bottom and bottom of box

OBJECTIVE-C
```
@property (nonatomic) CGFloat linksBottom
```

Discussion
CGFloat By default it is 10

Declared In
`SMInAppContentStyleOptions.h`

`marginBetweenLinks`

Set the constant between links margin - useful when there is two links that will be displayed

OBJECTIVE-C
```
@property (nonatomic) CGFloat marginBetweenLinks
```

Discussion

CGFloat By default it is 10

Declared In
`SMInAppContentStyleOptions.h`

`linkBorderWidth`

Set the border width for links

OBJECTIVE-C
```
@property (nonatomic) CGFloat linkBorderWidth
```

Discussion
CGFloat By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

`linkBorderColor`

Set the color of link Border

OBJECTIVE-C
```
@property (nonatomic) UIColor *linkBorderColor
```

Discussion
UIColor By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

`linkCornerRadius`

Set the corner radius for links

OBJECTIVE-C
```
@property (nonatomic) CGFloat linkCornerRadius
```

Discussion
CGFloat By default, it is not set

Declared In

`SMInAppContentStyleOptions.h`

`linkShadowColor`

Set the shadow color of all links

OBJECTIVE-C

```
@property (nonatomic) UIColor *linkShadowColor
```

Discussion
UIColor By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

`linkShadowOpacity`

Set the shadow opacity of all links

OBJECTIVE-C

```
@property (nonatomic) CGFloat linkShadowOpacity
```

Discussion
CGFloat By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

`linkShadowRadius`

Set the shadow radius of all links

OBJECTIVE-C

```
@property (nonatomic) CGFloat linkShadowRadius
```

Discussion
CGFloat By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

linkShadowOffset

Set the shadow offset of all links

OBJECTIVE-C
```
@property (nonatomic) CGSize linkShadowOffset
```

Discussion
CGSize By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

linkBackgroundColor

Set the background color of link

OBJECTIVE-C
```
@property (nonatomic) UIColor *linkBackgroundColor
```

Discussion
UIColor By default, it is whiteColor

Declared In
`SMInAppContentStyleOptions.h`

linkTextColor

Set the text color in link

OBJECTIVE-C
```
@property (nonatomic) UIColor *linkTextColor
```

Discussion
UIColor By default, it is whiteColor

Declared In
`SMInAppContentStyleOptions.h`

linkFont

Set the font of links

OBJECTIVE-C
```
@property (nonatomic) UIFont *linkFont
```

Discussion
UIFont By default, it is iOS default

Declared In
`SMInAppContentStyleOptions.h`

linkContentEdgeInsets

Set the link content edge inset

OBJECTIVE-C
```
@property (nonatomic) UIEdgeInsets linkContentEdgeInsets
```

Discussion
UIEdgeInsets By default, it is not set

Declared In
`SMInAppContentStyleOptions.h`

+ defaultStylingOptions

SMInAppContentStyleOptions constructor

OBJECTIVE-C
```
+ (instancetype)defaultStylingOptions
```

Return Value
SMInAppContentStyleOptions

Declared In
`SMInAppContentStyleOptions.h`

SMInAppContentURLViewController Class Reference

Inherits from	SMInAppContentViewController : UIViewController
Declared in	SMInAppContentURLViewController.h

Overview

a view controller for URL In App Content

SMInAppContentURLViewController :

+ viewControllerForCategory:

This will provide you a custom viewcontroller with one web view loaded with the url provided by an in app content for a specific category of an URL type

OBJECTIVE-C

```
+ (instancetype)viewControllerForCategory:(NSString *)category
```

Parameters

category	a NSString of the desired category of In App Content

Return Value
a SMInAppContentURLViewController

Discussion
The viewcontroller will take all available space in screen and will contain a close button if it is presented as it is. Otherwise the app will have to provide a UIContainerView which will contain the view controller

Declared In
`SMInAppContentURLViewController.h`

+ viewControllerForCategory:AndOptions:

This will provide you a custom viewcontroller with one web view loaded with the url provided by an in app content for a specific category of an URL type

OBJECTIVE-C

```
+ (instancetype)viewControllerForCategory:(NSString *)category AndOptions:
(SMInAppContentStyleOptions *)options
```

Parameters

category	a NSString of the desired category of In App Content
options	a SMInAppContentStyleOptions object allowing you to customise the in app content

Return Value

a SMInAppContentURLViewController

Discussion

The viewcontroller will take all available space in screen and will contain a close button if it is presented as it is. Otherwise the app will have to define a UIContainerView which will contain the view controller

Declared In

`SMInAppContentURLViewController.h`

SMInAppContentViewController Class Reference

Inherits from	UIViewController
Declared in	SMInAppContentViewController.h

Overview

parent class for SMInAppContentURLViewController, SMInAppContentHTMLViewController, SMInAppContentImageViewController

SMInAppContentViewController :

category

NSString containing the category of the SMInAppContentViewController

OBJECTIVE-C

```
@property (nonatomic, strong) NSString *category
```

Declared In

`SMInAppContentViewController.h`

isEmpty

bool set with true when the SMInAppContentViewController is empty

OBJECTIVE-C

```
@property (nonatomic) bool isEmpty
```

Declared In

`SMInAppContentViewController.h`

SMLink Class Reference

Inherits from	NSObject
Declared in	SMLink.h

Overview

SMLink :

`idButtonData`

NSString instance providing the id of the button

OBJECTIVE-C
```
@property (nonatomic, strong) NSString *idButtonData
```

Declared In
`SMLink.h`

`label`

NSString instance providing the label of the button

OBJECTIVE-C
```
@property (nonatomic, strong) NSString *label
```

Declared In
`SMLink.h`

`value`

NSString instance providing the value of the button

OBJECTIVE-C

```
@property (nonatomic, strong) NSString *value
```

Declared In

SMLink.h

type

The type (SMNotificationButtonType) of action that the button will execute.

OBJECTIVE-C

```
@property (nonatomic) SMNotificationButtonType type
```

See Also

- `SMNotificationButtonType for more information about each @property type`

Declared In

SMLink.h

SMManager Class Reference

Inherits from	NSObject
Declared in	SMManager.h

Overview

Start Library :

In order to start the library, please follow the steps bellow (will mainly happen in your UIApplication's delegate):

- Use startWithLaunchOptions:Setting: in your application:didFinishLaunchingWithOptions:
- Implement methods described in SMManager(RemoteNotification) in your UIApplication's delegate
- When building against iOS10 : Implement methods described in SMManager(UserNotification) in your UIApplication's delegate

Starting the library will not register for remote notification. Don't forget to call registerForRemoteNotification according to your application's need.

From there, your application is ready to handle all incoming remote-notifications.

SMManager Singleton :

This manager is the main interface third party developers will be using.

versionLib

The current version of the library

OBJECTIVE-C

```
@property (nonatomic) NSString *versionLib
```

Declared In

SMManager.h

+ sharedInstance

Singleton for SellMobileSDK Class which allow to access public SellMobileSDK methods and properties

OBJECTIVE-C

```
+ (instancetype)sharedInstance
```

Return Value
SMManager : singleton instance of SMManager class

Declared In
`SMManager.h`

- startWithLaunchOptions:Setting:

Mandatory method which allows sdk initialisation. To be included in application:didFinishLaunchingWithOptions: Make sure to provide the launchOptions dictionary

OBJECTIVE-C

```
- (void)startWithLaunchOptions:(NSDictionary *)launchOptions Setting:(SMManagerSetting *)setting
```

Parameters

launchOptions	NSDictionary instance indicating the reason the app was launched (if any). This dictionary is provided by application:didFinishLaunchingWithOptions
setting	mandatory SMManagerSetting instance to start-up the library

Discussion
This method is mandatory in order to start / initialise the library and should be called in application:didFinishLaunchingWithOptions:

See Also

- `SMManagerSetting`

Declared In
`SMManager.h`

- reloadSetting:

Optional + used for testing only. This method will re-configure the SMManager with the newly provided information

OBJECTIVE-C
```
- (void)reloadSetting:(SMManagerSetting *)setting
```

Parameters

setting	mandatory SMManagerSetting instance to start-up the library

Discussion

This is a handy API in case you would like to switch between two backend environements without rebuilding your target.

Warning: This API is provided for testing purposes only. Never use it in production. Make sure to re-enable any service after calling this API.

Declared In

`SMManager.h`

SMManagerSetting Class Reference

Inherits from	NSObject
Declared in	SMManagerSetting.h

Overview

Note about the SMManagerSetting object :

Creating an SMManagerSetting's instance is pretty straightforward as there's only one constructor for doing so. This sole constructor is sufficent to get you started.

Additional parameters described in this header will provide you with additional control and ***are all optional***.

Being user-friendly :

When the application is in foreground and receive a remote-notification, by default, the library will display it on the current visible UIViewController. This behaviour might be unwanted and may disturb the user if he appears to navigate in a different context. Should you want to prevent that behaviour and display the remote-notification shortly after (when the user will be in a more appropriate context), please follow these steps :

- Create an SMManagerSetting with the default constructor as usual.
- Set shouldDisplayRemoteNotification to FALSE.
- Start the library as usual [SMManager startWithLaunchOptions:Setting:]
- Listen to NSNotification-name: *kSMNotification_Event_DidReceiveRemoteNotification* (declared in SMNSNotification)

At this point, remote-notification are NOT displayed when the application is in foreground. (Other application's state are not affected). Then, displaying the remote-notification is up to the application and can be done at any time by :

- Retrieve the last remote-notification with [SMManager(RemoteNotification) retrieveLastRemotePushNotification]
- Display the notification according to its ID with [SMManager(RemoteNotification) displayNotificationID:]

Or, more straightforwardly :

- Display the last known remote notification by calling : [SMManager(RemoteNotification) displayLastReceivedRemotePushNotification]

IAM :

In-Application-Message-service is configurable using SMManagerSettingIAM which you'll inject using the API configureInAppMessageServiceWithSetting: A dedicated topic regarding this topic can be found in SMManager(InAppMessage)

IAC :

In-Application-Content-service is configurable using SMManagerSettingIAC which you'll inject using the API configureInAppContentServiceWithSetting: A dedicated topic regarding this topic can be found in SMManager(InAppContent)

SMManagerSetting :

This class allow you to configure the SMManager. Such instance must be created before starting the library.

shouldClearBadge

Once a new remote-notification is displayed, the badge is automaticly reseted. Should you want to handle this property yourself, you can set this property to FALSE before starting the library Default value is set to TRUE

OBJECTIVE-C
```
@property (nonatomic) BOOL shouldClearBadge
```

Declared In
`SMManagerSetting.h`

shouldDisplayRemoteNotification

Used to configure the remote-notification-display

OBJECTIVE-C
```
@property (nonatomic) BOOL shouldDisplayRemoteNotification
```

Discussion
When the app is active, once a new remote-notification is received, it is automatically diplayed on the device Should you want to prevent that behaviour, you can set this property to FALSE before starting the library. Default value is set to TRUE.

Warning: This property does not have an impact when app is open from a notification selected by user in the notification center or when [SMManager(RemoteNotification) displayNotificationID:] or

[SMManager(RemoteNotification) displayLastReceivedRemotePushNotification] are called. Once you set this value to TRUE, the application becomes responsable about displaying the remote-notification. (Make sure to read the header file of SMManagerSetting before doing so).

Declared In
`SMManagerSetting.h`

clearCacheIntervalValue

This value tells how often the SDK's cache mechanism should clear itself.

OBJECTIVE-C
```
@property (nonatomic) SMClearCache clearCacheIntervalValue
```

Discussion
Internally, each notification-messages has a life-span. Clearing the cache stands for deleting notification-messages with an expired life-span. In other words, only old notification-messages are deletes from the cache. More recent ones are kept in memory until their life-span expires and a new clearCache is called. By default, this value is set to kSMClearCache_Auto. Configuring this value highly depends of how frequently the application will query specific notification-messages. As if the application query a notification-message which is not in the cache anymore, it will automaticaly fetch it from the backend. In other words, it depends how often you call the API [SMManager(RemoteNotification) displayNotificationID:].

In a nutshell:

- If the application will never query [SMManager(RemoteNotification) displayNotificationID:], we recommend keeping this value to default.
- If the application use IAM-service, we recommend keeping this value to default.
- On the other hand, if the application abuse [SMManager(RemoteNotification) displayNotificationID:], we recommend selecting a value higher than the default one according to your application's need.

Warning: As soon as IAM-service is enabled, the SDK will consider kSMClearCache_Weekly as being the default value. Except if you explicitly overide the property. In 99% of the cases, you should not overide this property as the SDK is smart enough to handle the cache mechanism by itself.

Declared In
`SMManagerSetting.h`

+ settingWithUrl:ClientID:PrivateKey:

Default-mandatory constructor to start the SMManager shared-instance

OBJECTIVE-C
```
+ (id)settingWithUrl:(NSString *)urlName ClientID:(NSString *)clientID PrivateKey:(NSString
*)privateKey
```

Parameters

urlName	NSString instance representing the urlname of your backend.
clientID	NSString instance referencing the client's ID
privateKey	NSString instance containing a valid private-key used to secure requests

Return Value
SMManagerSetting new instance. SMFailure in case of error

Discussion
Warning: All these parameters are mandatory. If any of them is nil, the library won't start. Please contact our support to get valid configuration-setting.

Declared In
`SMManagerSetting.h`

– configureInAppMessageServiceWithSetting:

An invalid or nil setting is considered as an error and will not startUp the IAM-service. Don't forget to enable In App message according to application need by calling [SMManager(InAppMessage) enableInAppMessage:]

OBJECTIVE-C
```
- (void)configureInAppMessageServiceWithSetting:(SMManagerSettingIAM *)settingIAM
```

Parameters

settingIAM	The SMManagerSettingIAM instance containing the IAM desired configuration.

Discussion
This call is optional. It is not needed to successfully start the SDK. However, it is required to enable In-Application-Message. Please read SMManager(InAppMessage) for additional information.

Warning: An invalid or nil setting is considered as an error and will not startUp the IAM-service. Don't forget to enable In App message according to application need by calling [SMManager(InAppMessage) enableInAppMessage:]

Declared In
`SMManagerSetting.h`

– configureInAppContentServiceWithSetting:

An invalid or nil setting is considered as an error and will not startUp the IAC-service. Don't forget to enable In App content according to application need by calling [SMManager(InAppContent) enableInAppContent:]

OBJECTIVE-C

```
- (void)configureInAppContentServiceWithSetting:(SMManagerSettingIAC *)settingIAC
```

Parameters

settingIAC	The SMManagerSettingIAC instance containing the IAC desired configuration.

Discussion

This call is optional. It is not needed to successfully start the SDK. However, it is required to enable In-Application-Content. Please read SMManager(InAppContent) for additional information.

Warning: An invalid or nil setting is considered as an error and will not startUp the IAC-service. Don’t forget to enable In App content according to application need by calling [SMManager(InAppContent) enableInAppContent:]

Declared In

`SMManagerSetting.h`

– configureLocationService

To use location service you will need to have PlotProject.framework in your app.

OBJECTIVE-C

```
- (void)configureLocationService
```

Discussion

This call is optional. It is not needed to successfully start the SDK. However, it is the first step to enable Location service. It is the first step to inform the sdk that geo location services are going to be used by the

Warning: To use location service you will need to have PlotProject.framework in your app.

Declared In

`SMManagerSetting.h`

SMManagerSettingIAC Class Reference

Inherits from	NSObject
Declared in	SMManagerSettingIAC.h

Overview

This class allow you to configure the In-App-Content service. Such instance must be created before starting the library.

+ settingWithBackgroundFetchOnly

Constructor to be used in order to create the SMManagerSettingIAC instance

OBJECTIVE-C

```
+ (instancetype)settingWithBackgroundFetchOnly
```

Discussion
use this constructor to enable background-mode only. The OS will refresh automaticly the IAC based on how often the user interacts with the application

Warning: If background-fetch is not enabled in Application's Capabilities, the IAC-service will not start. See SMManager(InAppContent) for additional information.

Declared In
`SMManagerSettingIAC.h`

+ settingWithRefreshType:

Constructor to be used in order to create the SMManagerSettingIAC instance

OBJECTIVE-C

```
+ (instancetype)settingWithRefreshType:(SMInAppRefreshType)refreshType
```

Parameters

refreshType	The type of refresh to consider when the application is in foreground

Discussion

Use this constructor should you want to perform periodic refresh when the application is in foreground-state only. For enabling backgroundState, use settingWithRefreshType:ShouldPerformBackgroundFetch: instead

Declared In

`SMManagerSettingIAC.h`

+ settingWithRefreshType:ShouldPerformBackgroundFetch:

Constructor to be used in order to create the SMManagerSettingIAC instance

OBJECTIVE-C

```
+ (instancetype)settingWithRefreshType:(SMInAppRefreshType)refreshType
ShouldPerformBackgroundFetch:(BOOL)shouldPerformBackgroundFetch
```

Parameters

refreshType	The type of refresh to consider when the application is in foreground
shouldPerformBackgroundFetch	If set to TRUE, it will activate UIApplication-BackGround-Fetch-mode automaticly

Discussion

This constructor provides you with more control on foreground / background fetch. Should you want to restrict to only one fetch-mode, feel free to use other constructors.

Declared In

`SMManagerSettingIAC.h`

SMManagerSettingIAM Class Reference

Inherits from	NSObject
Declared in	SMManagerSettingIAM.h

Overview

This class allow you to configure the In-App-Message service. Such instance must be created before starting the library.

+ settingWithBackgroundFetchOnly

Constructor to be used in order to create the SMManagerSettingIAM instance

OBJECTIVE-C

```
+ (instancetype)settingWithBackgroundFetchOnly
```

Discussion
use this constructor to enable background-mode only. The OS will refresh automaticly the IAM based on how often the user interacts with the application

Warning: If background-fetch is not enabled in Application's Capabilities, the IAM-service will not start. See SMManager(InAppMessage) for additional information.

Declared In
`SMManagerSettingIAM.h`

+ settingWithRefreshType:

Constructor to be used in order to create the SMManagerSettingIAM instance

OBJECTIVE-C

```
+ (instancetype)settingWithRefreshType:(SMInAppRefreshType)refreshType
```

Parameters

refreshType	The type of refresh to consider when the application is in foreground

Discussion

Use this constructor should you want to perform periodic refresh when the application is in foreground-state only. For enabling backgroundState, use settingWithRefreshType:ShouldPerformBackgroundFetch: instead

Declared In

`SMManagerSettingIAM.h`

+ settingWithRefreshType:ShouldPerformBackgroundFetch:

Constructor to be used in order to create the SMManagerSettingIAM instance

OBJECTIVE-C

```
+ (instancetype)settingWithRefreshType:(SMInAppRefreshType)refreshType
ShouldPerformBackgroundFetch:(BOOL)shouldPerformBackgroundFetch
```

Parameters

refreshType	The type of refresh to consider when the application is in foreground
shouldPerformBackgroundFetch	If set to TRUE, it will activate UIApplication-BackGround-Fetch-mode automaticly

Discussion

This constructor provides you with more control on foreground / background fetch. Should you want to restrict to only one fetch-mode, feel free to use other constructors.

Declared In

`SMManagerSettingIAM.h`

SMMessage Class Reference

Inherits from	NSObject
Declared in	SMMessage.h

Overview

This Class is provided as a root Class and should not be used.

messageSM

NSString instance providing a brief description of the message

OBJECTIVE-C

```
@property (nonatomic, strong) NSString *messageSM
```

Declared In

`SMMessage.h`

SMNotificationButtonData Class Reference

Inherits from	SMLink : NSObject
Declared in	SMNotificationButtonData.h

Overview

This class is used to wrap informations about a notification button.

Additional information provided in SMManager

SMSuccess Class Reference

Inherits from	SMMessage : NSObject
Declared in	SMSuccess.h

Overview

This class is used to return a successfull action

SMSuccess:

SMClearCache Constants Reference

Declared in	SMClearCache.h

SMClearCache

Enumeration used to define how often the SDK’s cache should automaticaly clear itself

SMClearCache :

Definition

```
typedef NS_ENUM(NSInteger, SMClearCache ) {
   kSMClearCache_Auto,
   kSMClearCache_None,
   kSMClearCache_Week,
   kSMClearCache_Month,
   kSMClearCache_Quarter,
};
```

Constants

`kSMClearCache_Auto`

This is the default value.

Declared In `SMClearCache.h`.

`kSMClearCache_None`

This explicitely disable the SDK-cache mechanism

Declared In `SMClearCache.h`.

`kSMClearCache_Week`

Clear the cache weekly

Declared In `SMClearCache.h`.

`kSMClearCache_Month`

Clear the cache each month

Declared In `SMClearCache.h`.

`kSMClearCache_Quarter`

Clear the cache every three months

Declared In `SMClearCache.h`.

Declared In

`SMClearCache.h`

SMContentAlignment Constants Reference

Declared in	SMContentAlignment.h

SMContentAlignment

SMContentAlignment :

Definition

```
typedef NS_ENUM(NSInteger, SMContentAlignment ) {
   kSMAlignLeft,
   kSMAlignRight,
   kSMAlignCenter,
};
```

Constants

`kSMAlignLeft`
: The content will be left-aligned.

  Declared In `SMContentAlignment.h`.

`kSMAlignRight`
: The content will be right-aligned.

  Declared In `SMContentAlignment.h`.

`kSMAlignCenter`
: The content will be centered.

  Declared In `SMContentAlignment.h`.

Declared In

`SMContentAlignment.h`

SMDisplayMode Constants Reference

Declared in	SMDisplayMode.h

SMDisplayMode

SMDisplayMode :

Definition

```
typedef NS_ENUM(NSInteger, SMDisplayMode ) {
   kSMDisplayMode_Unknown = -1,
   kSMDisplayMode_OnlyOnce = 0,
   kSMDisplayMode_UntilReplaced = 1,
};
```

Constants

`kSMDisplayMode_Unknown`
    This explicitely sets the displayMode to unknown

    Declared In `SMDisplayMode.h`.

`kSMDisplayMode_OnlyOnce`
    Display only once

    Declared In `SMDisplayMode.h`.

`kSMDisplayMode_UntilReplaced`
    Always display until replaced

    Declared In `SMDisplayMode.h`.

Declared In
`SMDisplayMode.h`

SMInAppContentType Constants Reference

Declared in	SMInAppContentType.h

SMInAppContentType

SMInAppContentType :

Definition

```
typedef NS_OPTIONS(NSInteger, SMInAppContentType ) {
   kSMInAppContentType_Unknown = -2,
   kSMInAppContentType_HTML = 1,
   kSMInAppContentType_Url = 2,
   kSMInAppContentType_Image = 3,
};
```

Constants

`kSMInAppContentType_Unknown`
: In App content of unknown type.

  Declared In `SMInAppContentType.h`.

`kSMInAppContentType_HTML`
: In App content of HTML type.

  Declared In `SMInAppContentType.h`.

`kSMInAppContentType_Url`
: In App content of URL type.

  Declared In `SMInAppContentType.h`.

`kSMInAppContentType_Image`
: In App content of Image type.

  Declared In `SMInAppContentType.h`.

Declared In

`SMInAppContentType.h`

SMInAppRefreshType Constants Reference

Declared in	SMInAppRefreshType.h

SMInAppRefreshType

SMInAppRefreshType :

Definition

```
typedef NS_ENUM(NSInteger, SMInAppRefreshType ) {
   kSMIA_RefreshType_None,
   kSMIA_RefreshType_Minutely,
   kSMIA_RefreshType_Hourly,
   kSMIA_RefreshType_Daily,
};
```

Constants

`kSMIA_RefreshType_None`

This explicitely disable the In App fetch mecahnism

Declared In `SMInAppRefreshType.h`.

`kSMIA_RefreshType_Minutely`

Allow to fetch In App Minutely (to be used only in development)

Declared In `SMInAppRefreshType.h`.

`kSMIA_RefreshType_Hourly`

Allow to fetch In App hourly

Declared In `SMInAppRefreshType.h`.

`kSMIA_RefreshType_Daily`

Allow to fetch In App Daily

Declared In `SMInAppRefreshType.h`.

Declared In

`SMInAppRefreshType.h`

SMLocationAuthorisationStatus Constants Reference

Declared in	SMLocationAuthorisationType.h

SMLocationAuthorisationStatus

Location authorisation status.

Definition

```
typedef NS_ENUM(NSInteger, SMLocationAuthorisationStatus ) {
   kSMLocationAuthorisationStatus_Unknown,
   kSMLocationAuthorisationStatus_Refused,
   kSMLocationAuthorisationStatus_GrantedInUse,
   kSMLocationAuthorisationStatus_GrantedAlways,
};
```

Constants

`kSMLocationAuthorisationStatus_Unknown`
The authorisation satus is unknown.

Declared In `SMLocationAuthorisationType.h`.

`kSMLocationAuthorisationStatus_Refused`
The authorisation status is rejected by user or impossible

Declared In `SMLocationAuthorisationType.h`.

`kSMLocationAuthorisationStatus_GrantedInUse`
The authorisation status is OK when the application is in-use

Declared In `SMLocationAuthorisationType.h`.

`kSMLocationAuthorisationStatus_GrantedAlways`
The authorisation status is OK for all application’s state

Declared In `SMLocationAuthorisationType.h`.

Declared In

`SMLocationAuthorisationType.h`

SMLocationAuthorisationType Constants Reference

Declared in	SMLocationAuthorisationType.h

SMLocationAuthorisationType

Location authorisation type.

Definition

```
typedef NS_ENUM(NSInteger, SMLocationAuthorisationType ) {
   kSMLocationAuthorisationType_InUse,
   kSMLocationAuthorisationType_Always,
};
```

Constants

`kSMLocationAuthorisationType_InUse`

This value asks user permission to monitor their position only when the application is being used. Highly encouraged by Apple for efficiency purposes.

Declared In `SMLocationAuthorisationType.h`.

`kSMLocationAuthorisationType_Always`

This value asks user permission to monitor their position for all application's state. Use this option only if your application needs it.

Declared In `SMLocationAuthorisationType.h`.

Declared In

`SMLocationAuthorisationType.h`

SMLogLevel Constants Reference

Declared in	SMLog.h

SMLogLevel

Enumeration type for the log granularity

Definition

```
typedef NS_OPTIONS(NSInteger, SMLogLevel ) {
   kSMLogLevel_None = 0,
   kSMLogLevel_Info = 1 < < 1,
   kSMLogLevel_Warning = 1 < < 2,
   kSMLogLevel_Error = 1 < < 3,
   kSMLogLevel_HTTPCall = 1 < < 4,
   kSMLogLevel_Location = 1 < < 5,
   kSMLogLevel_All = 0 xFF,
};
```

Constants

`kSMLogLevel_None`
: No log printed at all. This is the suggested log-level for release.

  Declared In `SMLog.h`.

`kSMLogLevel_Info`
: Default log-entry. Basically inform user when library starts / ends.

  Declared In `SMLog.h`.

`kSMLogLevel_Warning`
: Only warning messages are printed

  Declared In `SMLog.h`.

`kSMLogLevel_Error`
: Only Error messages are being printed

  Declared In `SMLog.h`.

`kSMLogLevel_HTTPCall`
: Print only HTTP-requests stuff

  Declared In `SMLog.h`.

`kSMLogLevel_Location`

  Print only location-requests stuff

  Declared In `SMLog.h`.

`kSMLogLevel_All`

  Print everything. Do not use for release!!!

  Declared In `SMLog.h`.

Declared In

`SMLog.h`

SMNotificationButtonType Constants Reference

Declared in	SMNotificationButtonType.h

SMNotificationButtonType

This enumeration declares all known buttun-type

Definition

```
typedef NS_ENUM(NSInteger, SMNotificationButtonType ) {
   kSMNotificationButtonType_Unknown = -1,
   kSMNotificationButtonType_Simple = 0,
   kSMNotificationButtonType_OpenPhoneCall = 1,
   kSMNotificationButtonType_OpenSms = 2,
   kSMNotificationButtonType_OpenMail = 3,
   kSMNotificationButtonType_OpenBrowser = 4,
   kSMNotificationButtonType_OpenApplication = 5,
   kSMNotificationButtonType_RateApplication = 6,
   kSMNotificationButtonType_CustomActionBroadcastEvent = 7,
   kSMNotificationButtonType_Return_Text = 8,
   kSMNotificationButtonType_Return_Photo = 9,
   kSMNotificationButtonType_Return_TextAndPhoto = 10,
   kSMNotificationButtonType_Passbook = 11,
   kSMNotificationButtonType_DeepLink = 13,
};
```

Constants

`kSMNotificationButtonType_Unknown`

Any received button-type not in this enumeration type will be considered as unknown

Declared In `SMNotificationButtonType.h`.

`kSMNotificationButtonType_Simple`

A button-type that will have no action but when clicked will send back button value to the platform

Declared In `SMNotificationButtonType.h`.

`kSMNotificationButtonType_OpenPhoneCall`

A button-type that will open the Phone application with a ready to use number to dial

Declared In `SMNotificationButtonType.h`.

`kSMNotificationButtonType_OpenSms`

A button-type that will open the sms application with a ready to dial sms

Declared In `SMNotificationButtonType.h`.

`kSMNotificationButtonType_OpenMail`
A button-type that will open the mail application

Declared In `SMNotificationButtonType.h`.

`kSMNotificationButtonType_OpenBrowser`
Button that will open a ready to use safari-browser

Declared In `SMNotificationButtonType.h`.

`kSMNotificationButtonType_OpenApplication`
Button that will open a third party application

Declared In `SMNotificationButtonType.h`.

`kSMNotificationButtonType_RateApplication`
Button-type which will allow application rating. This will behave similarly to kSMNotificationButtonType_OpenApplication In Android terminology, this notion is called “Store”

Declared In `SMNotificationButtonType.h`.

`kSMNotificationButtonType_CustomActionBroadcastEvent`
Button that will trigger a notification (broadcast in Android terminology) inside the application for any interested listener. You may register in your application to a specific event from NSNotificationCenter. The notification is broadcasted as soon as the push is received. Your back-end team should inform you about the notification-name. No parameters are currently supported.

Declared In `SMNotificationButtonType.h`.

`kSMNotificationButtonType_Return_Text`
Button-type which will allow user to provide back a media-type as answer Media-type of kind Text

Declared In `SMNotificationButtonType.h`.

`kSMNotificationButtonType_Return_Photo`
Button-type which will allow user to provide back a media-type as answer Media-type of kind Picture

Declared In `SMNotificationButtonType.h`.

`kSMNotificationButtonType_Return_TextAndPhoto`
Button-type which will allow user to provide back a media-type as answer Media-type of kind Text + Picture

Declared In `SMNotificationButtonType.h`.

`kSMNotificationButtonType_Passbook`
Button-type which will allow user to add pkpass file inside wallet app

Declared In `SMNotificationButtonType.h`.

`kSMNotificationButtonType_DeepLink`

Button-type which will allow user depplink inside specific page in the app

Declared In `SMNotificationButtonType.h`.

Declared In

`SMNotificationButtonType.h`

SMManager(DataTransaction) Category Reference

Declared in	SMManager+DataTransaction.h

Overview

This category will help you to retrieve or send specific information to the selligent platform.

SMManager+DataTransaction :

– sendDeviceInfo

Deprecated method

OBJECTIVE-C
```
- (void)sendDeviceInfo
```

Discussion
Warning: Deprecated method

In order to send device informations to the back-end you can use

- sendDeviceInfo:(SMDeviceInfos*)infos;

Declared In
`SMManager+DataTransaction.h`

– sendDeviceInfo:

This method allow you to send an event with pre-defined devices informations to the back-end

OBJECTIVE-C
```
- (void)sendDeviceInfo:(SMDeviceInfos *)deviceInfos
```

Parameters

deviceInfos	SMDeviceInfos object

Discussion

The informations to be sent will be set in a SMDeviceInfos object. Each property of the created object will be a data that will be provided to the back-end. For additional informations you can refer to SMDevicesInfos.

This call can be done at any time after starting the library and if the properties of SMDeviceInfos object has not changed from a previous call, the event will not be sent to avoid platform unnecessarily overload.

Declared In

`SMManager+DataTransaction.h`

SMManager(InAppContent) Category Reference

Declared in	SMManager+InAppContent.h

Overview

Introduction :

In-Application-Content (IAC) is an optional service which will retrieve messages from the back-end each time the application enters foregrounds at specific frequencies and if connection is available.

Once new messages were retrieved, the library notifies the application about it.

Each IAC is is from a specific type [SMInAppContentType] and is also linked to a category defined by yourself

Implementation :

In a nutshell, activate the IAC-service is a one step process:

- Create an SMManagerSettingIAC instance and inject it in SMManagerSetting with [SMManagerSetting configureInAppContentServiceWithSetting:]

In order to be notified about new IAC, the application must register to correct notification *kSMNotification_Event_DidReceiveInAppContent* (Please read SMNSNotification for additional information). This notification will provide you with the number of IAC's by category. Please be aware that it's the unique application's chance to capture and store that information.

Displaying IAC :

- With the SDK view controllers:

Each IAC is from a specific type for a specific category.

Selligent SDK provide a specific view controller for each type of in app content

- SMInAppContentHTMLViewController for IAC of type kSMInAppContentType_HTML
- SMInAppContentURLViewController for IAC of type kSMInAppContentType_Url
- SMInAppContentImageViewController for IAC of type kSMInAppContentType_Image

You can check each of this object for more information about how to use them.

All this view controller can also be customised with the use of SMInAppContentStyleOptions.

Once the sdk has provided you with the correct view controller

```
SMInAppContentURLViewController* vc = [SMInAppContentURLViewController
viewControllerForCategory:@"anycategory"];
```

You can call showSMController:InContainerView:OfParentViewController: if you expect to display the In App Content in a UIContainerView that is defined in your app :

```
[[SMManager sharedInstance] showSMController:vc InContainerView:_containerView
OfParentViewController:self];
```

Or you can present it to be displayed in full screen mode :

```
[self presentViewController:vc animated:YES completion:nil];
```

Be aware that if your UIContainerView is defined in storyboard and that no category has been provided to it you will need to inform it with prepareForSegue:sender:

```
-(void) prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
if([segue.identifier isEqualToString:@"YourSegue"]){ _sminappcontentviewcontroller =
segue.destinationViewController; [self.sminappcontentviewcontroller
setCategory:@"anycategory"]; } }
```

- With your own view controllers

In this case you can call getInAppContentsForCategory:Type: or getInAppContentsForCategory:Type:Max:

Those methods will present you an NSArray of SMInAppContentMessage with all (or a maximum number, precised by the Max parameter) IAC for a category and for a type.

If you decide to use this way of interacting with IAC it is important that you keep in mind that you will be responsible of the display of the content and you will have to call to setInAppContentAsSeen: whenever an InAppContent is showed to the user and to executeLinkAction:InAppContent: whenever user interact with an action link of the in app content.

Fetching modes :

IAC may be retrieved from two different modes corresponding to the application’s state:

- Foreground-fetch – When the application is in foreground.
- Background-fetch – When the application is in background.

Each of these modes is **optional**. In other words, you can use one mode or the other, or even both at the same time. Choosing the adequate mode depends on the application’s need and can be managed by the application’s developper. Off course, to allow the SDK to retrieve IAC as fast as possible, we recommand using both modes at the same time.

Following documentation explains how to activate each mode:

Fetching IAC in foreground :

In order to retrieve IAC while the application is running, make sure to do the folllowing:

- Create and configure an SMManagerSettingIAC instance accordingly.
- Provide the created SMManagerSettingIAC instance to SMManager before starting the library

Fetching IAC in background :

Initially, this mode has been added as a complementary-option to the foreground-mode. However, it can be used as a single-fetch-mode if fits best your application's need.

To understand better how often the OS will execute the background-fetch, make sure to read the following Apple-documentation

In order to support this mode, make sure to :

- Create the SMManagerSettingIAC instance accordingly.
- Provide the created SMManagerSettingIAC instance to SMManager before starting the library.
- In the application's target, enable the following: Capabilities > Background Modes > Background Fetch
- Implement performFetchWithCompletionHandler: in UIApplication's delegate (under delegate method application:performFetchWithCompletionHandler:)

SMManager+InAppContent :

`- showSMController:InContainerView:OfParentViewController:`

Call when SDK has already provided you with a SMInAppContentViewController (of any type : SMInAppContentHTMLViewController , SMInAppContentImageViewController or SMInAppContentURLViewController) and you want to display it in a UIContainerView defined anywhere in your app.

OBJECTIVE-C

```
- (void)showSMController:(SMInAppContentViewController *)smViewController InContainerView:(UIView *)containerView OfParentViewController:(UIViewController *)parentViewController
```

Parameters

smViewController	a SMInAppContentViewController
containerView	the UIContainerView in which In App Content will be displayed
parentViewController	the parent UIViewController of your UIContainerView

Discussion

The viewcontroller will take all available space in the UIContainerView

Declared In
`SMManager+InAppContent.h`

- getInAppContentsForCategory:Type:

This will return an array of In App Contents

OBJECTIVE-C
```
- (NSArray *)getInAppContentsForCategory:(NSString *)category Type:(SMInAppContentType)type
```

Parameters

category	NSString the category for which you want your In App Contents
type	An SMInAppContentType - Url, Html or image

Return Value
returns an NSArray of SMInAppContentMessage

Discussion
All the IAC will be retrieved

Declared In
`SMManager+InAppContent.h`

- getInAppContentsForCategory:Type:Max:

This will return an array of In App Contents

OBJECTIVE-C
```
- (NSArray *)getInAppContentsForCategory:(NSString *)category Type:(SMInAppContentType)type Max:
(int)max
```

Parameters

category	An NSString the category for which you want your In App Contents
type	An SMInAppContentType - Url, Html or image
max	An int

Return Value
returns an NSArray of SMInAppContentMessage

Discussion
This overload allows you to define the max number of IAC to retrieve

Declared In
`SMManager+InAppContent.h`

- setInAppContentAsSeen:

This method will mark an IAC as viewed, save it in the cache and send the Open event to the server

OBJECTIVE-C
```
- (void)setInAppContentAsSeen:(SMInAppContentMessage *)inAppContent
```

Parameters

inAppContent	an SMInAppContentMessage object

Discussion
If the display mode is set to 0 (display once), the IAC will be discarded from the cache and will not be provided to you anymore with getInAppContentsForCategory:Type: or getInAppContentsForCategory:Type:Max:

Declared In
`SMManager+InAppContent.h`

- executeLinkAction:InAppContent:

This method must be called whenever a user has clicked on a link that you manage to display

OBJECTIVE-C
```
- (void)executeLinkAction:(SMLink *)link InAppContent:(SMInAppContentMessage *)inAppContent
```

Parameters

link	a SMLink object
inAppContent	a SMInAppContentMessage object

Discussion
This will allow the sdk to inform the services that a link has been clicked and to process the action associated with the link

Declared In
`SMManager+InAppContent.h`

- performIACFetchWithCompletionHandler:

This will allow the SDK to fetch the IAC when the OS will allow so.

OBJECTIVE-C

```
- (void)performIACFetchWithCompletionHandler:(void ( ^ ) ( UIBackgroundFetchResult
))completionHandler
```

Parameters

completionHandler	The block-completion to be processed. Provided by the delegate call

Discussion

To be included in application:performFetchWithCompletionHandler:

Warning: Make sure to enable background-fetch in the application's capabilities

Declared In

`SMManager+InAppContent.h`

SMManager(InAppMessage) Category Reference

Declared in	SMManager+InAppMessage.h

Overview

Introduction :

In-Application-Message (IAM) is an optional service which will **automatically** retrieve messages from the back-end at specific frequencies. Once new messages were retrieved, the library notifies the application about it. Then, the application may display any IAM as a usual notification.

The retrieved IAM share the exact same format as the remote-notification. However, they are not live messages and, therefore should NOT be considered as an alternative to remote-notification. Remote-push-notificaitons are almost live messages that are pushed to devices. While IAM are non-live-messages that the SDK fetch at specific intervals according to provided fetch-mode.

Implementation :

In a nutshell, activate the IAM-service is a two steps process:

- First, create an SMManagerSettingIAM instance and inject it in SMManagerSetting with [SMManagerSetting configureInAppMessageServiceWithSetting:]
- Then, enable the IAM-service with enableInAppMessage:

However, additional steps might be required according to desired fetching-mode. Each of these steps are explained in further details bellow.

In order to be notified about new IAM, the application must register to correct notification *kSMNotification_Event_DidReceiveInAppMessage* (Please read SMNSNotification for additional information). This notification will provide you with the IAM's unique ID. Please be aware that it's the unique application's chance to capture and store that information.

Displaying IAM :

As IAM and remote-notification share the same format, they are both displayed using the same APIs. Please read the documentation in SMManager(RemoteNotification) to know how to display any kind of notification.

Fetching modes :

IAM may be retrieved from two different modes corresponding to the application's state:

- Foreground-fetch – When the application is in foreground.
- Background-fetch – When the application is in background.

Each of these modes is **optional**. In other words, you can use one mode or the other, or even both at the same time. Choosing the adequate mode depends on the application's need and can be managed by the application's developper. Off course, to allow the SDK to retrieve IAM as fast as possible, we recommand using both modes at the same time.

Following documentation explains how to activate each mode:

Fetching IAM in foreground :

In order to retrieve IAM while the application is running, make sure to do the folllowing:

- Create and configure an SMManagerSettingIAM instance accordingly.
- Provide the created SMManagerSettingIAM instance to SMManager before starting the library
- Enable In App message by calling enableInAppMessage: when your application needs it.

Fetching IAM in background :

Initially, this mode has been added as a complementary-option to the foreground-mode. However, it can be used as a single-fetch-mode if fits best your application's need.

To understand better how often the OS will execute the background-fetch, make sure to read the following Apple-documentation

In order to support this mode, make sure to :

- Create the SMManagerSettingIAM instance accordingly.
- Provide the created SMManagerSettingIAM instance to SMManager before starting the library.
- In the application's target, enable the following: Capabilities > Background Modes > Background Fetch
- Implement performFetchWithCompletionHandler: in UIApplicaiton's delegate (under delegate method application:performFetchWithCompletionHandler:)
- Enable In App message by calling enableInAppMessage: when your application needs it.

SMManager+InAppMessage :

`- enableInAppMessage:`

Call this API in order to enable / disable the IAM-service according to your application’s need.

OBJECTIVE-C
```
- (void)enableInAppMessage:(BOOL)shouldEnable
```

Parameters

shouldEnable	TRUE will enable IAM. FALSE will disable it.

Discussion
Most application will call this API right after starting the SDK. However, it make no harm to call it later on when user trigger, for instance, a UISwitch.

Warning: Before enabling IAM-service, make sure to :

- Start the SDK. Enabling the IAM-service before starting the SDK will have no effect.
- Configure correctly the IAM-service via configureInAppMessageServiceWithSetting:

Declared In
`SMManager+InAppMessage.h`

- performIAMFetchWithCompletionHandler:

This will allow the SDK to fetch the IAM when the OS will allow so.

OBJECTIVE-C
```
- (void)performIAMFetchWithCompletionHandler:(void ( ^ ) ( UIBackgroundFetchResult
))completionHandler
```

Parameters

completionHandler	The block-completion to be processed. Provided by the delegate call

Discussion
To be included in application:performFetchWithCompletionHandler:

Warning: Make sure to enable background-fetch in the application’s capabilities

Declared In
`SMManager+InAppMessage.h`

SMManager(Location) Category Reference

Declared in	SMManager+Location.h

Overview

Introduction :

Location is an optional service which will **automatically** provide with user's location-information. Location-information are provided at regular intervals according to provided configuration.

This information is not granted! Users may decide to share their location and then, few days afterwards, stop sharing it. The SDK will *automatically* inform the back-end of any permission changes.

Implementation :

In order to correctly start the location-service, please follow these steps :

- Add a key-value pair in your info-plist-file (Additional details in the section below).
- Optional step depending on the value of the enableOnFirstRun in your plotconfig.json file: if set to false you will have to enableGeoLocation
- Finally, you can call requestLocationAuthorisation: to ask user permissions according to your application's need.

Accessing User Data :

Requesting user's authorisation is a mandatory step to access their location-information. The dialogue asking for this permission will only be displayed once in the application life cycle.

If user denies sharing location-information, the application can NOT display the dialogue a 2nd time. Thus, he'll have to manually access the Settings-application and enable location sharing from there.

Configuring Info-Plist :

For the reason stated above, it is extremely important to explain to users the exact reason why to access their location information. A key-value pair explaining that reason must be added to the application's Info-Plist-file.

The **Key** depends on SMLocationAuthorisationType used :

- For kSMLocationAuthorisationType_InUse use **NSLocationWhenInUseUsageDescription**
- For kSMLocationAuthorisationType_Always use **NSLocationAlwaysUsageDescription** or **NSLocationAlwaysAndWhenInUseUsageDescription** since iOS11

The **Value** represents the string-sentence that will be displayed in the alert.

To know more about Plist-configuration, please see Apple documentation. Also, Apple does provide few guidelines for displaying the string-sentence.

SMManager+Location :

– currentAuthorisationStatus

If this value is kSMLocationAuthorisationStatus_Refused –> Users must activate location from Settings-Application. Alerts are only displayed once!

OBJECTIVE-C

```
- (SMLocationAuthorisationStatus)currentAuthorisationStatus
```

Return Value
The current authorisation status for this application.

Discussion
Please check SMLocationAuthorisationStatus to understand each case.

Warning: If this value is kSMLocationAuthorisationStatus_Refused –> Users must activate location from Settings-Application. Alerts are only displayed once!

Declared In
SMManager+Location.h

– requestLocationAuthorisation:

Ask user’s permission to share his location

OBJECTIVE-C

```
- (void)requestLocationAuthorisation:(SMLocationAuthorisationType)type
```

Parameters

type	The requested location-authorisation-type check SMLocationAuthorisationType to understand each type.

Discussion
This call will display an alert requesting user’s location-information. The alert will only be displayed once. However, it may no harm to make this call several times. Call this API according to your application’s need.

Declared In
`SMManager+Location.h`

– enableGeoLocation

Enable geolocation services

OBJECTIVE-C
```
- (void)enableGeoLocation
```

Discussion
This optional call will enable geolocation services at sdk level. it is independent of iOS location authorisation If you use plotproject for geolocation it will be mandatory to call it if you have set enableOnFirstRun to false in the plotconfig.json Call this API according to your application’s need.

Declared In
`SMManager+Location.h`

– disableGeoLocation

Disable geolocation services

OBJECTIVE-C
```
- (void)disableGeoLocation
```

Discussion
This optional call will disable geolocation services at sdk level. it is independent of iOS location authorisation Call this API according to your application’s need.

Declared In
`SMManager+Location.h`

– isGeoLocationEnabled

check geolocation services status

OBJECTIVE-C
```
- (BOOL)isGeoLocationEnabled
```

Discussion

This optional call will inform you if geolocation services at sdk level are enabled. it is independent of iOS location authorisation

Declared In

`SMManager+Location.h`

SMManager(Log) Category Reference

Declared in	SMManager+Log.h

Overview

This category will help you debug the library. Please check SMLogLevel for all available possibilities.

Should you want to get back to us, please set logLevel to kSMLogLevel_All and provide with console logs.

SMManager+Log :

– applyLogLevel:

Set the log level of the library console

OBJECTIVE-C
```
- (void)applyLogLevel:(SMLogLevel)logLevel
```

Parameters

logLevel	SMLogLevel enumeration type. Default = kSMLoLevel_None

Discussion
This is an optional setting that may help you debug the library calls. This call can be done at any time (before or after starting the library). However, in order to avoid missing any error log, we recommand setting this value before starting the library.

Warning: It is developer's responsability to enable log-level in Debug or release mode. No distinction are being aplied by the library. For obvious performance reason, it is always recommended to turn log off in release mode.

Declared In
`SMManager+Log.h`

SMManager(RemoteNotification) Category Reference

Declared in	SMManager+RemoteNotification.h

Overview

This category contains the basic step-by-step implementation to get you started. Make sure to read the category SMManager(SilentPush) to learn more about background-remote-notification.

Handling Remote Notifications:

In order to receive remote-notification from the back-end, all the following methods must be included in you application's delegate:

- didRegisterForRemoteNotificationsWithDeviceToken:
- didRegisterUserNotificationSettings:
- didFailToRegisterForRemoteNotificationsWithError:
- didReceiveRemoteNotification:

Finally, make sure to call registerForRemoteNotification according to your application's need.

Receiving Remote Notifications:

When a remote-notification is received, the library will automatically display a custom UIViewController. Should you want to prevent this behaviour, feel free to configure the SMManagerSetting accordingly before starting the SMManager.

Before displaying the remote-notification's UIVIewController, the library will broadcast an NSNotification offering the application a chance to pause any ongoing heavy task. The same principle is applied before dismissing the UIVIewController providing the application the opportunity to start again the paused heavy-task.

Finally, should you want to know when the user interact with UIViewController's control, an NSNotification is also posted providing information about the selected element. For additionnal information about NSNotification processing and handling, please check SMNSNotification

Displaying notification :

The application can display any notification based on its ID using the API displayNotificationID: These IDs can be retrieved from broadcasted NSNotification. (Please read SMNSNotification for additional information).

A convenient method is provided to display the last received remote-notification using displayLastReceivedRemotePushNotification

SMManager+RemoteNotification :

- registerForRemoteNotification

Mandatory method which allows notification registration

OBJECTIVE-C
```
- (void)registerForRemoteNotification
```

Discussion
This API will display a dialog asking user’s permission for remote-notifications (when app is launched the very 1st time). Often, this call is added right after startWithLaunchOptions:Setting: However, you may call this API later in your code according to your application need. Just remember that this call is mandatory to receive remote-notifications

Warning: If your device has already been registered to remote-notifications by your application or a third-party framework, then this call is not mandatory.

Declared In
`SMManager+RemoteNotification.h`

- unregisterForRemoteNotification

Use this API to unregister the current device. In other words, the device will not receive any remote-notification from our backend server anymore.

OBJECTIVE-C
```
- (void)unregisterForRemoteNotification
```

Discussion
Warning: This does NOT call unregisterForRemoteNotifications on the sharedApplication instance. Therefore, the application can still receive third-party remote-notifications.

Declared In
`SMManager+RemoteNotification.h`

- didRegisterForRemoteNotificationsWithDeviceToken:

Mandatory API to be included in application:didRegisterForRemoteNotificationsWithDeviceToken:

OBJECTIVE-C
```
- (void)didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
```

Parameters

deviceToken	A string that identifies the device to APNs.

Discussion
This method is mandatory to handle properly all notifications

Declared In
`SMManager+RemoteNotification.h`

– didRegisterUserNotificationSettings:

Deprecated method - replaced by didRegisterUserNotificationSettings; Mandatory API to be included in application:didRegisterUserNotificationSettings

OBJECTIVE-C
```
- (void)didRegisterUserNotificationSettings:(UIUserNotificationSettings *)notificationSettings
```

Parameters

notificationSettings	The user notification settings that are available to your app.

Discussion
Warning: Deprecated method - replaced by didRegisterUserNotificationSettings; Mandatory API to be included in application:didRegisterUserNotificationSettings

This method confirms the type of notification the user would like to receive.

Declared In
`SMManager+RemoteNotification.h`

– didRegisterUserNotificationSettings

Mandatory API to be included in application:didRegisterUserNotificationSettings

OBJECTIVE-C
```
- (void)didRegisterUserNotificationSettings
```

Discussion
This method confirms the type of notification the user would like to receive.

Declared In
`SMManager+RemoteNotification.h`

– didFailToRegisterForRemoteNotificationsWithError:

Mandatory API to be included in application:didFailToRegisterForRemoteNotificationsWithError

OBJECTIVE-C
```
- (void)didFailToRegisterForRemoteNotificationsWithError:(NSError *)error
```

Parameters

error	An NSError object that encapsulates information why registration did not succeed. Provided by the delegate call

Discussion
Called when the user has deactivated remote-notification or if any other error happen.

Declared In
`SMManager+RemoteNotification.h`

– didReceiveRemoteNotification:

Mandatory API to be included in application:didReceiveRemoteNotification Handle and display remote notification when application is in foreground

OBJECTIVE-C
```
- (void)didReceiveRemoteNotification:(NSDictionary *)userInfo
```

Parameters

userInfo	An NSDictionary that contains information related to the remote notification. Provided by the delegate call

Discussion
This method is not mandatory anymore if you implement didReceiveRemoteNotification:fetchCompletionHandler: For additional information about background-remote-notifications, please check SMManager(SilentPush) for further details.

Declared In
`SMManager+RemoteNotification.h`

- displayNotificationID:

Display a notification based on its ID

OBJECTIVE-C

```
- (void)displayNotificationID:(NSString *)idNotification
```

Parameters

idNotification	NSString instance referencing the unique notification's ID

Discussion

Basically, the application may store notification's IDs and then display them according to its need. In this context, the word "notification" stands for either a remote-notification or an InAppMessage. This feature has initially been added to allow applications to display remote-notifications at any time (not directly when the push is received). Then, it has been extended to display In-App-Messages. For additional information about IAM, please read SMManager(InAppMessage).

Declared In

SMManager+RemoteNotification.h

- displayLastReceivedRemotePushNotification

Retrieve and display the last known notification.

OBJECTIVE-C

```
- (void)displayLastReceivedRemotePushNotification
```

Discussion

Basically, This API is a helpher which combine both retrieveLastRemotePushNotification and displayNotificationID: It only focuses on remote-notification. Not on IAM. At this stage, only the very last remote-notification can be recovered, previous ones are automaticly overriden. To learn more about this API, please read documentation in SMManagerSetting, more particularly [SMManagerSetting shouldDisplayRemoteNotification]

Declared In

SMManager+RemoteNotification.h

- retrieveLastRemotePushNotification

Retrieve information about the last received remote-notification

OBJECTIVE-C

```
- (NSDictionary *)retrieveLastRemotePushNotification
```

Return Value

NSDictionary instance containing basic information about the last push, nil if no push was received so far.

Discussion

This is a convinient method to quickly retrieve the last remote-notificaiton known by the device. At this stage, only the very last remote-notification can be recovered, previous ones are automaticly overriden. To learn more about this API, please read documentation in SMManagerSetting, more particularly [SMManagerSetting shouldDisplayRemoteNotification]

Declared In

`SMManager+RemoteNotification.h`

SMManager(SMEvent) Category Reference

Declared in	SMManager+SMEvent.h

Overview

Sending events :

Sending any set of data to the back-end can be done with the API sendSMEvent:

Event type :

Few default events are already available for you to be used. They all inhirit from SMEvent and are configurable through their constructors. At the time of this writing, they default provided events are :

- SMEventUserLogin
- SMEventUserLogout
- SMEventUserRegistration
- SMEventUserUnregistration

Custom events :

Simplest case is to create an instance of SMEvent. Then, inject your data in it (Code example bellow).

Also, you can subclass from default provided event-type or even create your own sub-classes of events.

The library will keep sending events to the backend as far as they inhirit from SMEvent.

Injecting custom data in events :

Any information can be appended to an event and sent to your back-end. This is basically done by creating a dictionary containing your data and injecting it as in the example bellow.

@code // Dictionary with your custom data NSDictionary *dictMyCustomData = @{@“MyKey”: @“MyValue”}; // Create the event SMEvent* event = [SMEvent eventWithDictionary:dictMyCustomData]; // Sent the event to the back-end [[SMManager sharedInstance] sendSMEvent:event]; @endcode

The exemple above is considered as a custom event. The same principle can be applied to any event-type subclasses stated above or to your own subclasses of SMEvent.

SMManager+SMEvent :

– sendSMEvent:

Send an event to the Selligent platform

OBJECTIVE-C

```
- (void)sendSMEvent:(SMEvent *)event
```

Parameters

event	An SMEvent object containing your event

Discussion

Should you want to track the event’s response, please check SMEvent

Declared In

`SMManager+SMEvent.h`

SMManager(SilentPush) Category Reference

Declared in	SMManager+SilentPush.h

Overview

Optionally, you can support silent-remote-notificaiton which will not render anything on the device. To know more about this topic, please visit the Apple documentation.

Even if you’re not planning to use silent-pushes, we recommand enabling this service in your application anyway. As using this service will also improve rendering time for the usual (non-silent) remote-notifications.

Implementation :

In a nutshell, you should do the following :

- In the application’s target, enable the following: Capabilities > Background Modes > Remote Notifications
- Removing previous call to didReceiveRemoteNotification: (see last point of “Start library”)
- Implement didReceiveRemoteNotification:fetchCompletionHandler: in UIApplication’s delegate.

SMManager+SilentPush :

– didReceiveRemoteNotification:fetchCompletionHandler:

Mandatory API to be included in application:didReceiveRemoteNotification:fetchCompletionHandler Handle and display the received notification according to different application state.

OBJECTIVE-C

```
- (void)didReceiveRemoteNotification:(NSDictionary *)userInfo fetchCompletionHandler:(void ( ^ )
( UIBackgroundFetchResult ))completionHandler
```

Parameters

userInfo	An NSDictionary that contains information related to the remote notification. Provided by the delegate call
completionHandler	The block-completion to be processed after the download. Provided by the delegate call

Discussion

It is recommended to use this API over didReceiveRemoteNotification: as it handles silent-remote-notificaitons.

Warning: You must enable “Remote notifications” in your application’s Capabilities in order to use this API. If this capability is not usefull to your application, you must use didReceiveRemoteNotification: instead.

Declared In

`SMManager+SilentPush.h`

– didReceiveRemoteNotification:fetchCompletionHandler:ForceResultFetch:

See didReceiveRemoteNotification:fetchCompletionHandler:

OBJECTIVE-C

```
- (void)didReceiveRemoteNotification:(NSDictionary *)userInfo fetchCompletionHandler:(void ( ^ )
( UIBackgroundFetchResult ))completionHandler ForceResultFetch:
(UIBackgroundFetchResult)resultFetch
```

Parameters

userInfo	An NSDictionary that contains information related to the remote notification. Provided by the delegate call
completionHandler	The block-completion to be processed after the download. Provided by the delegate call
resultFetch	The enumeration that might be overiden by application if needed

Discussion

This API is provided in order to force the fetch result to a specific value. Use it only if your application need to overide the service. Otherwise, please use didReceiveRemoteNotification:fetchCompletionHandler:

Warning: You must enable “Remote notifications” in your application’s Capabilities in order to use this API. If this capability is not usefull to your application, you must use didReceiveRemoteNotification: instead.

Declared In

`SMManager+SilentPush.h`

SMManager(StyleOptions) Category Reference

Declared in	SMManager+StyleOptions.h

Overview

Allows you to customise the appearance of all the In App Content containers

Implementation :

- First create a SMInAppContentStyleOptions object instance
- load it with loadStyleOptions:

SMManager+StyleOptions :

- loadStyleOptions:

This will allow you to load your custom SMInAppContentStyleOptions object

OBJECTIVE-C
```
- (void)loadStyleOptions:(SMInAppContentStyleOptions *)options
```

Parameters

options	a SMInAppContentStyleOptions object

Declared In
`SMManager+StyleOptions.h`

- resetStyleOptions

Reset style options to default one

OBJECTIVE-C

```
- (void)resetStyleOptions
```

Declared In

`SMManager+StyleOptions.h`

SMManager(UserNotification) Category Reference

Declared in	SMManager+UserNotification.h

Overview

In addition to the implementation of category SMManager(RemoteNotification), this category will supply a support of iOS 10 and usage of UserNotifications framework

Make sure to read the category SMManager(RemoteNotification) to learn more about remote-notification for support of iOS version 8 and 9 and to correctly register for remote notification.

If you build against iOS 10+ and you use UserNotifications framework we then recommend you to implement those 2 methods of UNUserNotificationCenterDelegate in your appdelegate:

- SMManager didReceiveNotificationResponse:
- SMManager didReceiveNotificationResponse:withCompletionHandler:
- SMManager willPresentNotification:
- SMManager willPresentNotification:withCompletionHandler:

Register your appdelegate to the UNUserNotificationCenter by adding the following two lines in the didFinishLaunchingWithOptions delegate.

```
UNUserNotificationCenter *center = [UNUserNotificationCenter
currentNotificationCenter];

center.delegate = self;
```

Notification extensions

In this category you will also find the support for notification extensions target (content and service)

Init extensions (to be used for each extensions that will be added to your project):

- SMManager startExtensionWithSetting:

Notification content extension :

- SMManager didReceiveNotification:

Notification service extension :

- SMManager didReceiveNotificationRequest:
- SMManager didReceiveNotificationRequest:withContentHandler:

- SMManager serviceExtensionTimeWillExpire

SMManager+UserNotifications :

- didReceiveNotificationResponse:

Mandatory API, when building against iOS 10+ and using UserNotifications framework, to be included in userNotificationCenter:didReceiveNotificationResponse:withCompletionHandler. Handle and display remote notification.

OBJECTIVE-C

```
- (void)didReceiveNotificationResponse:(UNNotificationResponse *)response
```

Parameters

response	A UNNotificationResponse that contains information about the notification and the interaction the user has done with it. Provided by the delegate call

Declared In

`SMManager+UserNotification.h`

- didReceiveNotificationResponse:withCompletionHandler:

Mandatory API, when building against iOS 10+ and using UserNotifications framework, to be included in userNotificationCenter:didReceiveNotificationResponse:withCompletionHandler Handle and display remote notification.

OBJECTIVE-C

```
- (void)didReceiveNotificationResponse:(UNNotificationResponse *)response withCompletionHandler:
(void ( ^ ) ( void ))completionHandler
```

Parameters

response	A UNNotificationResponse that contains information about the notification and the interaction the user has done with it. Provided by the delegate call
completionHandler	A completion that will be called.

Declared In

`SMManager+UserNotification.h`

– willPresentNotification:

Mandatory API, when building against iOS 10+ and using UserNotifications framework, to be included in userNotificationCenter:willPresentNotification:withCompletionHandler Handle the remote notification when app is in foreground.

OBJECTIVE-C

```
- (void)willPresentNotification:(UNNotification *)notification
```

Parameters

notification	A UNNotification that contains information about the notification.

Discussion

This allows the SDK to inform the services that the Selligent push has been received.

Calling this method does not call any CompletionHandler. You will have to call the completion handler with your preferred option.

For example you can call inside this delegate : completionHandler(UNNotificationPresentationOptionAlert);

Declared In

`SMManager+UserNotification.h`

– willPresentNotification:withCompletionHandler:

Mandatory API, when building against iOS 10+ and using UserNotifications framework, to be included in userNotificationCenter:willPresentNotification:withCompletionHandler Handle the remote notification when app is in foreground and call the completionHandler(UNNotificationPresentationOptionAlert) by default.

OBJECTIVE-C

```
- (void)willPresentNotification:(UNNotification *)notification withCompletionHandler:(void ( ^ )
( UNNotificationPresentationOptions options ))completionHandler
```

Parameters

notification	A UNNotification that contains information about the notification.
completionHandler	A Completion handler that will be called with UNNotificationPresentationOptionAlert by default.

Discussion

This allows the SDK to inform the services that the Selligent push has been received.

You don’t need to call the completion handler in the delegate anymore.

Declared In

SMManager+UserNotification.h

- startExtensionWithSetting:

Mandatory method which allows sdk initialisation when building against iOS 10+ and using Notification Extensions . To be included in didReceiveNotification: when implementing Notification content extension and/or in didReceiveNotificationRequest:withContentHandler: when implementing Notification service extension

OBJECTIVE-C
```
- (void)startExtensionWithSetting:(SMManagerSetting *)setting
```

Parameters

setting	mandatory SMManagerSetting instance to start-up the library

Discussion
This method is mandatory in order to start / initialise the library and should be called in in didReceiveNotification: when implementing Notification content extension and/or in didReceiveNotificationRequest:withContentHandler: when implementing Notification service extension

See Also

- `SMManagerSetting`

Declared In
SMManager+UserNotification.h

- didReceiveNotification:

Optional API, when building against iOS 10+ and using a Notification Content Extension target, to be included in NotificationViewController didReceiveNotification: Handle the push action buttons that may be present in the selligent notification payload

OBJECTIVE-C
```
- (void)didReceiveNotification:(UNNotification *)notification
```

Parameters

notification	A UNNotification that contains information about the notification.

Discussion
This allows the SDK to display the action buttons in the push notification outside of the app.

The category that must be added to Info.plist of the Notification Content Extension target is mandatory and must be SELLIGENT_BUTTON.

Declared In
`SMManager+UserNotification.h`

– didReceiveNotificationRequest:

Optional API, when building against iOS 10+ and using a Notification Service Extension target, to be included in NotificationService didReceiveNotificationRequest:withContentHandler: Handle the selligent payload before displaying it to the user

OBJECTIVE-C
```
- (UNMutableNotificationContent *)didReceiveNotificationRequest:(UNNotificationRequest *)request
```

Parameters

request	A UNNotificationRequest that contains the original notification request.

Return Value
UNMutableNotificationContent the updated content of the payload.

Discussion
This allows the SDK to decrypt the payload before displaying it to the user if you have activated the encryption of push.

you can use this method if you have decided to trigger the block to execute with the modified content by yourself otherwise if you want the sdk to manage all steps please use SMManager didReceiveNotificationRequest:withContentHandler:

Declared In
`SMManager+UserNotification.h`

– didReceiveNotificationRequest:withContentHandler:

Optional API, when building against iOS 10+ and using a Notification Service Extension target, to be included in NotificationService didReceiveNotificationRequest:withContentHandler: Handle the selligent payload before displaying it to the user

OBJECTIVE-C
```
- (void)didReceiveNotificationRequest:(UNNotificationRequest *)request withContentHandler:(void (
^ ) ( UNNotificationContent *))contentHandler
```

Parameters

request	A UNNotificationRequest that contains the original notification request.
contentHandler	The block to execute with the modified content

Discussion
This allows the SDK to decrypt the payload before displaying it to the user if you have activated the encryption of push.

you can use this method if you want the sdk to manage the display of the notification after teh decryption has been processed

Declared In
`SMManager+UserNotification.h`

- serviceExtensionTimeWillExpire

Optional API, when building against iOS 10+ and using a Notification Service Extension target, to be included in NotificationService serviceExtensionTimeWillExpire Tells the sdk that the extension is about to be terminated.

OBJECTIVE-C
```
- (void)serviceExtensionTimeWillExpire
```

Discussion
this method is to be implemented only if you have implemented SMManager didReceiveNotificationRequest:withContentHandler:

Declared In
`SMManager+UserNotification.h`

SMCompletionBlockSuccess Block Reference

Declared in	SMBlock.h

Block Definition

SMCompletionBlockSuccess

@typedef type of block that will be triggered when an event has been succesfully sent

```
typedef void (^SMCompletionBlockSuccess) (SMSuccess *success)
```

Declared In
`SMBlock.h`

SMCompletionBlockFailure Block Reference

Declared in	SMBlock.h

Block Definition

SMCompletionBlockFailure

@typedef type of block that will be triggered when an event has failed to be sent

```
typedef void (^SMCompletionBlockFailure) (SMFailure *failure)
```

Declared In

`SMBlock.h`