CrystalExpress Integration Guide

Table of content

1. Before SDK integration

- Make sure you have get CrystalExpress.plist from Intowow. It will look like this.
  - If you don't have `Crystal_Id`, please contact Intowow to request one for you app.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList
-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Crystal_Id</key>
    <string>2a317bd038f742a082ee284b478a8e37</string>
</dict>
</plist>
```

2. How to integrate CrystalSDK lib to project ?

2.1 Using Cocoapods

- We strongly recommand you to use Cocoapods to integrate with CrystalExpress.
- Add the following code in Podfile `pod "CrystalExpressSDK", '~> 1.1'`
- `pod update` or `pod install`
- Open workspace that pod generate for you, you're ready to use CrystalExpress
- Here's a sample project

2.2 Manual integration

1. In project build phases "Link Binary With Libraries", add libCrystalSDK-release-x.x.x.a static library
2. Add header file to your project
3. Make sure you have the following frameworks added in Build phases
   - Securty.framework
   - CFNetwork.framework
   - MessageUI.framework
   - MobileCoreServices.framework
   - SystemConfiguration.framework
   - AdSupport.framework
   - libz.dylib
   - libc++.dylib
   - CoreTelephony.framework
   - CoreMedia.framework
   - libsqlite3.dylib
   - AVFoundation.framework
   - libicucore.dylib
4. Add `-ObjC` in TARGETS -> Build Settings -> Linking -> Other Linker Flags
5. Add the following files to your project -> Supporting Files

    - kinesis-2013-12-02.json
    - cognito-identity-2014-06-30.json
    - sts-2011-06-15.json
    - CrystalExpress.plist
6. You can now start using CrystalExpress lib.

3. CrystalExpress APIs

- We describe CrystalExpress APIs in detail here, start at general APIs, and following with different types of ADs.

3.1 General AD serving APIs

I2WAPI.h

```objc
// initialize I2WAPI with whether to enable verbose log
+ (void)initWithVerboseLog:(BOOL)enableVerbose;

// return whether CrystalExpress is ready for AD serving
+ (BOOL)isAdServing;

// tell CrystalExpress it's time to refresh ads (including delete useless ad creatives)
+ (void)refreshI2WAds;

// trigger background fetch task
+ (void)triggerBackgroundFetchOnSuccess:(void (^)())success
                                 onFail:(void (^)())fail
                              onNoData:(void (^)())noData;

// set active placement, this affect AD prefetch priority
+ (void)setActivePlacement:(NSString *)placement;

#pragma mark - track API
// track customized event
+ (void)trackCustomEventWithType:(NSString *)type props:(NSDictionary *)props;

// update geolocation infomation
+ (void)updateUserLastLocation:(NSDictionary *)location;

#pragma mark - deep link
// handle CrystalExpress related deeplink url
+ (void)handleDeepLinkWithUrl:(NSURL *)url sourceApplication:(NSString *)sourceApplication;

#pragma mark - AD related
+ (void)getSplashADWithPlacement:(NSString *)placement
                           place:(int)place
                            type:(CESplashType)splashType
                         onReady:(void (^)(ADView *adView, BOOL fitsMultiOffer))ready
                       onFailure:(void (^)(NSError *error))failure;

+ (void)getBannerADWithPlacement:(NSString *)placement
                         onReady:(void (^)(ADView *))ready
                       onFailure:(void (^)(NSError *))failure;

+ (void)getStreamADWithPlacement:(NSString *)placement
                       helperKey:(NSString *)helperKey
                           place:(int)place
                         onReady:(void (^)(ADView *adView))ready
                       onFailure:(void (^)(NSError *error))failure
              onPullDownAnimation:(void (^)(UIView *))animation;

+ (void)getContentADWithPlacement:(NSString *)placement
                            isPreroll:(BOOL)isPreroll
                              onReady:(void (^)(ADView *))ready
                            onFailure:(void (^)(NSError *))failure
                   onPullDownAnimation:(void (^)(UIView *))animation;
```

3.2 Splash AD

- We provided a helper class to make integration more easier, via SplashADHelper, you can request different format of Splash ADs
- SplashADHelper will call delegate function and return a ready `SplashADInterfaceViewController` for you to present.

SplashADHelper.h

```
@protocol SplashADHelperDelegate <NSObject>
@required
// SplashADHelper call this function while the SplshAD viewcontroller is ready to present
- (void)SplashADDidReceiveAd:(NSArray *)ad viewController:(SplashADInterfaceViewController *)
vc;

// SplashADHelper call this function while encounter error, such as fail to find available s
plash ADs
- (void)SplashADDidFailToReceiveAdWithError:(NSError *)error viewController:(SplashADInterfa
ceViewController *)vc;
@end

// We have predefined different modes for Splash AD viewcontroller
// CE_SPLASH_MODE_HYBRID       --> You might get a multi/single offer Splash AD
// CE_SPLASH_MODE_MULTI_OFFER  --> You will get a multioffer Splash AD
// CE_SPLASH_MODE_SINGLE_OFFER --> You will get a singleoffer Splash AD
typedef NS_ENUM(NSUInteger, CESplashMode) {
    CE_SPLASH_MODE_UNKNOWN,
    CE_SPLASH_MODE_HYBRID,
    CE_SPLASH_MODE_MULTI_OFFER,
    CE_SPLASH_MODE_SINGLE_OFFER,
};

@interface SplashADHelper : NSObject
@property (nonatomic, weak) id<SplashADHelperDelegate> delegate;

// request SplashAD with placement name and mode
- (void)requestSplashADWithPlacement:(NSString *)placement mode:(CESplashMode)splashMode;
- @end
```

SplashADInterfaceViewController

- The splash AD view controller are the member of SplashADInterfaceViewController.

```
@class SplashADInterfaceViewController;

// By register the SplashADViewControllerDelegat, you can get different stage of splash even
ts
@protocol SplashADViewControllerDelegate <NSObject>
@optional
- (void)SplashAdWillDismissScreen:(SplashADInterfaceViewController *)vc;
- (void)SplashAdWillPresentScreen:(SplashADInterfaceViewController *)vc;
- (void)SplashAdDidDismissScreen:(SplashADInterfaceViewController *)vc;
- (void)SplashAdDidPresentScreen:(SplashADInterfaceViewController *)vc;
@end
```

3.3 Content AD

- Utilize ContentADHelper class to request content AD and manage AD
- Init helper by giving a AD placement name.
- `preroll` can prepare 1 Article AD in advance. Use this function in `ViewDidLoad` or other pre-stage, giving helper more time to prepare a AD.
- Once the positon of ad is decided, call `setScrollOffsetWithKey` to update the AD's scroll offset.
- Call `requestADWithContentId:(NSString *)articleId` with article_Id to get a AD UIView.
- Update the scroll view bounds while scroll view did scroll like the following code.

```
#pragma mark - scrollview delegate
- (void)scrollViewDidScroll:(UIScrollView *)scrollView
{
    [_contentADHelper updateScrollViewBounds:[scrollView bounds] withKey:_articleId];
}
```

- Check ad should start / stop while your UI is in stable status, such as scroll view end decerlating, view controller appear/disappear.

```
- (void)viewDidDisappear:(BOOL)animated
{
    [super viewDidDisappear:animated];
    [_contentADHelper checkAdStartWithKey:_artId ScrollViewBounds:CGRectZero];
}

- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
    [_contentADHelper checkAdStartWithKey:_artId ScrollViewBounds:_scrollView.bounds];
}

- (void)scrollViewDidEndDragging:(UIScrollView *)scrollView willDecelerate:(BOOL)decelerate
{
    if (decelerate == NO) {
        [_contentADHelper checkAdStartWithKey:_articleId ScrollViewBounds:[scrollView bounds]
];
    }
}

- (void)scrollViewDidEndDecelerating:(UIScrollView *)scrollView
{
    [_contentADHelper checkAdStartWithKey:_articleId ScrollViewBounds:[scrollView bounds]];
}
```

- If content AD formats including pulldown card, set `onPullDownAnimation` block to update scroll view while pulldown animating is happened.

```
// set animation for pulldown card, need to update the module offset below the AD
__weak typeof(self) weakSelf = self;
[_contentADHelper setOnPullDownAnimation:^(UIView *view) {
   float offset = weakSelf.AdOffset + view.frame.size.height;
   [weakSelf updateBottomContentFromOffset:offset];
}];
```

Complete API

```
#pragma mark - ContentADHelper.h

@interface ContentADHelper : NSObject
// pulldown animation block
@property (nonatomic, copy) void (^onPullDownAnimation)(UIView *adView);

- (instancetype)initWithPlacement:(NSString *)placement;
- (void)preroll;
- (ADView *)requestADWithContentId:(NSString *)articleId;
- (void)setScrollOffsetWithKey:(NSString *)key offset:(int)offset;
- (void)checkAdStartWithKey:(NSString *)key ScrollViewBounds:(CGRect)bounds;
- (void)updateScrollViewBounds:(CGRect)bounds withKey:(NSString *)key;
@end
```

3.4 Stream AD

- Utilize StreamADHelper class to request stream AD and manage AD.
- Init helper by giving a AD placement name.
- Set delegate for the helper instance.
- `preroll` can prepare 1 stream AD in advance. Use this function in `ViewDidLoad` or other pre-stage, giving helper more time to prepare a AD.
- Call `updateVisiblePosition:(UITableView *)tableView` once the initial tableview source is ready.

```
- (void)viewDidLoad {
    .....

    [self prepareDataSource];
    [self.tableView reloadData];
    [_streamHelper updateVisiblePosition:[self tableView]];

   .....
}
```

- Set active status based on the view controller is showing to user.

```
[_streamHelper setActive:YES];
```

- Call `requestADAtPosition:(int)position` cell position to get a AD UIView.
- Sync helper while `scrollViewDidScroll:(UIScrollView *)scrollView` event happened.

```
- (void)scrollViewDidScroll:(UIScrollView *)scrollView
{
    [_streamHelper scrollViewDidScroll:scrollView tableView:[self tableView]];
}
```

- Call `scrollViewStateChanged` if scrollview status change, this give helper a chance to check AD should trigger start/stop event.

```objc
- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
    [_streamHelper setActive:YES];
    [_streamHelper scrollViewStateChanged];
}

- (void)viewDidDisappear:(BOOL)animated
{
    [super viewDidDisappear:animated];
    [_streamHelper setActive:NO];
    [_streamHelper scrollViewStateChanged];
}

- (void)scrollViewDidEndDragging:(UIScrollView *)scrollView willDecelerate:(BOOL)decelerate
{
    if (decelerate == NO) {
        [_streamHelper scrollViewStateChanged];
    }
}

- (void)scrollViewDidEndDecelerating:(UIScrollView *)scrollView
{
    [_streamHelper scrollViewStateChanged];
}
```

- Implement `StreamADHelperDelegate` functions
- `- (int)onADLoaded:(UIView *)adView atPosition:(int)position` need to update table view to insert AD cell. Return the real position inserted in table view, or -1 if fail.
- While in preroll, there's no need to insert AD in another main loop

```
- (int)onADLoaded:(UIView *)adView atPosition:(int)position isPreroll:(BOOL)isPreroll
{
    // Don't place ad at the first place!!
    position = MAX(1, position);
    if ([_dataSource count] >= position) {
        if (isPreroll) {
            NSMutableDictionary *adDict = [[NSMutableDictionary alloc] init];
            [adDict setObject:[NSNumber numberWithInt:adView.bounds.size.height] forKey:@"he
ight"];

            NSArray *indexPathsToAdd = @[[NSIndexPath indexPathForRow:position inSection:0]];

            [[self tableView] beginUpdates];
            [_dataSource insertObject:adDict atIndex:position];
            [[self tableView] insertRowsAtIndexPaths:indexPathsToAdd
                                    withRowAnimation:UITableViewRowAnimationNone];
            [[self tableView] endUpdates];
        } else {
            dispatch_async(dispatch_get_main_queue(), ^(){
                NSMutableDictionary *adDict = [[NSMutableDictionary alloc] init];
                [adDict setObject:[NSNumber numberWithInt:adView.bounds.size.height] forKey:
@"height"];

                NSArray *indexPathsToAdd = @[[NSIndexPath indexPathForRow:position inSection:
0]];
                [[self tableView] beginUpdates];
                [_dataSource insertObject:adDict atIndex:position];
                [[self tableView] insertRowsAtIndexPaths:indexPathsToAdd
                                        withRowAnimation:UITableViewRowAnimationNone];
                [[self tableView] endUpdates];
            });
        }
        return position;
    } else {
        return -1;
    }
}
```

- If stream AD formats including pulldown card, set `onADAnimation:(UIView *)adView atPosition:(int)position` block to update scroll view while pulldown animating is happened.

```
- (void)onADAnimation:(UIView *)adView atPosition:(int)position
{
    [UIView animateWithDuration:1.0 delay:0.0 options:UIViewAnimationOptionAllowUserInteract
ion animations:^{
        [[self tableView] beginUpdates];
        [[_dataSource objectAtIndex:position] setObject:[NSNumber numberWithInt:adView.bounds
.size.height] forKey:@"height"];
        [[self tableView] endUpdates];
    } completion:^(BOOL finished) {

    }];
}
```

- Implement `checkIdle` to tell helper whether it is a good timing to start/stop AD, trigger AD in stable state will improve the user experience.

```
- (BOOL)checkIdle
{
    return (![[self tableView] isDecelerating] && ![[self tableView] isDragging]);
}
```

Complete API

```objc
#pragma mark - StreamADHelper.h

@class ADView;
@protocol StreamADHelperDelegate <NSObject>
// callback function while the stream ad is ready at the request position
- (int)onADLoaded:(UIView *)adView atPosition:(int)position;

// callback on pull down animation happen
- (void)onADAnimation:(UIView *)adView atPosition:(int)position;

// callback to check whether the view is in idle state
- (BOOL)checkIdle;
@end

@interface StreamADHelper : NSObject
@property (nonatomic, assign) BOOL isActiveSection;
@property (nonatomic, weak) id<StreamADHelperDelegate> delegate;

// init helper with placement name
- (instancetype)initWithPlacement:(NSString *)placement;

// preroll will request a stream ad for future use
- (void)preroll;

// request stream ad at stream position
- (UIView *)requestADAtPosition:(int)position;

// update current table view visible cell
- (void)updateVisiblePosition:(UITableView *)tableView;

// get all loaded ads
- (NSOrderedSet *)getLoadedAds;

// force all loaded ad stop
- (void)stopADs;

// set helper active state, ad will only play in active helper
- (void)setActive:(BOOL)isActive;

// check whether the position is an AD
- (BOOL)isAdAtPos:(int)pos;

#pragma mark - event listener
// scroll view did scroll event hook
- (void)scrollViewDidScroll:(UIScrollView *)scrollView tableView:(UITableView *)tableView;

// trigger AD state change while scrollview state changed
- (void)scrollViewStateChanged;
@end
```

3.5 Flip AD

- Flip AD is one of the splash series AD.
- Utilize FlipDynamicADHelper class to request flip AD and manage AD.
- Init helper by giving a AD placement name.
- `setActive` to trigger helper prefetch current placement group's AD.
- Get flip AD view by calling `requestADAtPosition:(int)position`
- Call `onPageSelectedAtPositoin:(int)position` to trigger decision of AD start/stop, we suggest to call this function while your UI is in stable status to improve the UX.

```
- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
    [_flipADHelper onPageSelectedAtPositoin:(int)_curIndex];
}

- (void)viewDidDisappear:(BOOL)animated
{
    [super viewDidDisappear:animated];
    [_flipADHelper onPageSelectedAtPositoin:-1];
}

- (void)scrollViewDidEndDecelerating:(UIScrollView *)scrollView
{
    CGFloat pageWidth = CGRectGetWidth([scrollView frame]);
    NSUInteger page = floor((scrollView.contentOffset.x - pageWidth / 2) / pageWidth) + 1;
    [_flipADHelper onPageSelectedAtPositoin:(int)page];

}
```

Complete API

```
@interface FlipDynamicADHelper : NSObject
// initialize with placement name and page index
- (instancetype)initWithPlacement:(NSString *)placement
                        pageIndex:(NSUInteger)pageIndex;

// enable flip AD helper
- (void)setActive;

// request an AD with flip index position
- (UIView *)requestADAtPosition:(int)position;

#pragma mark - event listener
// decide whether AD start/stop on current pages
- (void)onPageSelectedAtPositoin:(int)position;
@end
```

- Call `onStop` to stop current AD while the view controller is disappear.

```
- (void)viewDidDisappear:(BOOL)animated
{
    [_flipAdHelper onStop];
}
```

3.6 Banner AD

- Utilize BannerADHelper class to request banner AD and manage AD.
- Init helper by giving a AD placement name.
- Get banner AD view by calling
  `- (void)requestADonReady:(void (^)(ADView *))ready onFailure:(void (^)(NSError *))failure`

```
[_bannerAdHelper requestADonReady:^(ADView *adView) {
    [self didReceiveBannerAd:adView];
} onFailure:^(NSError *error) {
    NSLog("Fail to get banner AD due to %@", error);
}];
```

- Call `onStart` to start AD.
- Call `onStop` to stop AD.

Complete API

```
@class ADView;
@interface BannerADHelper : NSObject

// initialize with placement name
- (instancetype)initWithPlacement:(NSString *)placement;

// request banner AD with ready/fail block
- (void)requestADonReady:(void (^)(ADView *))ready
               onFailure:(void (^)(NSError *))failure;

// onStop to stop current banner AD
- (void)onStop;

// onStart to start current banner AD
- (void)onStart;
@end
```

---

4. Register background task

- In `AppDelegate.m`, register a background task will allow CrystalExpress SDK able to fetch ads while app enter background mode.

```
- (void)applicationDidEnterBackground:(UIApplication *)application
{
    // register bgTask while enter background mode
    __block UIBackgroundTaskIdentifier bgTask = [application beginBackgroundTaskWithExpirati
onHandler:^{
        [application endBackgroundTask:bgTask];
        bgTask = UIBackgroundTaskInvalid;
    }];
}
```

5. Register background fetch

1. In project settings, Target -> Capabilities -> Turn Background Modes to **ON**, Check **Background Fetch**
2. In `AppDelegate.m` add function like the following code:

```
- (void)application:(UIApplication *)application performFetchWithCompletionHandler:(void (^)(
UIBackgroundFetchResult))completionHandler
{
    [I2WAPI triggerBackgroundFetchOnSuccess:^{
        completionHandler(UIBackgroundFetchResultNewData);
    } onFail:^{
        completionHandler(UIBackgroundFetchResultFailed);
    } onNoData:^{
        completionHandler(UIBackgroundFetchResultNoData);
    }];
}
```

6. AD Preview

In Demo App, you can utilize deeplink to do AD preview, sample url link like follows:
`crystalexpress://adpreview?adid={number}`

```
#pragma mark - deeplinking
- (BOOL)application:(UIApplication *)application
                      openURL:(NSURL *)url
    sourceApplication:(NSString *)sourceApplication
               annotation:(id)annotation
{
        NSLog(@"deep linking:");
        NSLog(@"  [url host] : %@", [url host]);
        NSLog(@"  [url path] : %@", [url path]);
        NSLog(@"  [sourceApplication] : %@", sourceApplication);
        [I2WAPI handleDeepLinkWithUrl:url sourceApplication:sourceApplication];
        return YES;
}
```

7. Tracking behavior

- We show CrystalExpress tracking message details in this section, inclidung the message meaning, the message send timing and its example message log
- We divide tracking message into some categories as the following
- Each category has several types of messages, represent different meaning

7.1 CATEGORY_APP (App Level Message)

- **REGISTER**
    - send while first time initialize CrystalExpress


```
{"time":1430892319244,"device_id":"5D03C597duck21E3duck4E2Fduck8D53duck2E4FABF5374D","crystal_id":"2a317bd038f742a082ee284b478a8e37","cat":"APP","nt":2,"type":"register","version":4,"props":{"ov":"8.2","av":"1.0.2","ot":1,"dm":"Simulator","sv":10010002,"mf":"Apple Inc.","idfa":"BA4EF51EFC2E44B89FDC07664427DB7A"}}
```


- **UPGRADE**
    - send while app version upgrade


```
{"time":1430892465825,"device_id":"5D03C597duck21E3duck4E2Fduck8D53duck2E4FABF5374D","crystal_id":"2a317bd038f742a082ee284b478a8e37","cat":"APP","nt":2,"type":"upgrade","version":4,"props":{"ov":"8.2","av":"1.0.2","ot":1,"dm":"Simulator","sv":10010002,"mf":"Apple Inc.","idfa":"BA4EF51EFC2E44B89FDC07664427DB7A"}}
```


- **OPEN**
    - send while app enter foreground


```
{"time":1430892674811,"device_id":"5D03C597duck21E3duck4E2Fduck8D53duck2E4FABF5374D","crystal_id":"2a317bd038f742a082ee284b478a8e37","cat":"APP","nt":2,"type":"open","version":4}
```


- **CLOSE**
    - send while app enter background or terminate


```
{"time":1430892719706,"device_id":"5D03C597duck21E3duck4E2Fduck8D53duck2E4FABF5374D","crystal_id":"2a317bd038f742a082ee284b478a8e37","cat":"APP","nt":2,"type":"close","version":4,"props":{"duration":45011}}
```


- **BACKGROUND_FETCH**
    - send while app is triggered by ios background fetch


```
{"time":1430892735224,"device_id":"5D03C597duck21E3duck4E2Fduck8D53duck2E4FABF5374D","crystal_id":"2a317bd038f742a082ee284b478a8e37","cat":"APP","nt":2,"type":"background_fetch","version":4}
```

7.2 CATEGORY_ADREQ (AD Request)

- **AD_REQUEST**
  - send while request an AD


```
{"time":1430892933014,"device_id":"5D03C597duck21E3duck4E2Fduck8D53duck2E4FABF5374D","crystal_id":"2a317bd038f742a082ee284b478a8e37","cat":"ADREQ","nt":2,"type":"ad_request","version":4,"props":{"requests":{"OPEN_SPLASH":{"1":1}}}}
```


7.3 CATEGORY_AD (AD Level Message)

- **FETCH**
  - send while successfully fetch an AD creatives from server


```
{"time":1430877242249,"device_id":"5D03C597duck21E3duck4E2Fduck8D53duck2E4FABF5374D","crystal_id":"2a317bd038f742a082ee284b478a8e37","cat":"AD","nt":2,"type":"fetch","version":4,"props":{"item_id":1176}}
```


- **IMPRESSION**
  - send while an AD is viewed by user


```
{"time":1430893382649,"device_id":"5D03C597duck21E3duck4E2Fduck8D53duck2E4FABF5374D","crystal_id":"2a317bd038f742a082ee284b478a8e37","cat":"AD","nt":2,"type":"impression","version":4,"props":{"item_id":1865,"place":1,"placement":"STREAM"}}
```


- **CLICK**
  - send while user click on AD engage area


```
{"time":1430893518315,"device_id":"5D03C597duck21E3duck4E2Fduck8D53duck2E4FABF5374D","crystal_id":"2a317bd038f742a082ee284b478a8e37","cat":"AD","nt":2,"type":"click","version":4,"props":{"item_id":1865,"place":1,"placement":"STREAM"}}
```


- **VIDEO_VIEW**
  - send while user had watched a video AD, or a video AD is play to the end.


```
{"time":1430893420791,"device_id":"5D03C597duck21E3duck4E2Fduck8D53duck2E4FABF5374D","crystal_id":"2a317bd038f742a082ee284b478a8e37","cat":"AD","nt":2,"type":"video_view","version":4,"props":{"place":1,"percentage":6,"engaged":false,"item_id":1818,"duration":1535}}
```


- **REMOVE**
  - send while SDK delete an AD's creative


```
{"time":1430877239787,"device_id":"5D03C597duck21E3duck4E2Fduck8D53duck2E4FABF5374D","crystal_id":"2a317bd038f742a082ee284b478a8e37","cat":"AD","nt":2,"type":"remove","version":4,"props":{"item_id":2332}}
```

- **MUTE**
    - send while user mute a video AD

```
{"time":1430893473329,"device_id":"5D03C597duck21E3duck4E2Fduck8D53duck2E4FABF5374D","crystal_id":"2a317bd038f742a082ee284b478a8e37","cat":"AD","nt":2,"type":"mute","version":4,"props":{"item_id":1865,"place":1,"placement":"STREAM"}}
```

- **UNMUTE**
    - send while user unmute a video AD

```
{"time":1430893471095,"device_id":"5D03C597duck21E3duck4E2Fduck8D53duck2E4FABF5374D","crystal_id":"2a317bd038f742a082ee284b478a8e37","cat":"AD","nt":2,"type":"unmute","version":4,"props":{"item_id":1865,"place":1,"placement":"STREAM"}}
```

- **REPLAY**
    - send while user click on video replay button

```
[TRCKER] {"time":1430893735319,"device_id":"5D03C597duck21E3duck4E2Fduck8D53duck2E4FABF5374D","crystal_id":"2a317bd038f742a082ee284b478a8e37","cat":"AD","nt":2,"type":"replay","version":4,"props":{"item_id":1799,"place":1,"placement":"OPEN_SPLASH"}}
```

8. Trouble shooting

- Request AD but nothing happened?
    - open the verbose log while initialize I2WAPI
    - check the log while request AD
- `[__NSArrayI enumFromString:]: unrecognized selector sent to instance 0x78e37970`
    - If you crash on log like this, add `-ObjC` in TARGETS -> Build Settings -> Linking -> Other Linker Flags

```
// this means your crystal_id is not correct, change it in CrystalExpress.plist
error:[Request failed: not found (404)], please reverify your crystal_id is set correct
```