Config

A Config instance is used to configure the DASH instance for use.

```
// A unique identifier for the DASH Foundation (provided by
DASH)
public String appId;
```

DASH

A DASH sharedInstance is the main class for interacting with the SDK. Use this instance to handle deep link URLs and to instantiate a view controller to show the DASH interface.

```
/**
 * Singleton use of DASH.
 *
 * @return Shared instance of DASH
 */
public static DASH getInstance()

/**
 * Initializes DASH. Call this once before any other methods.
 *
 * @param context A context
 * @param config An instance of Config
 */
public void startWithConfig(Context context, Config config)

/**
 * Sets the current user's email. Email is used to uniquely
identify a user in the DASH system.
 * Email is cached locally by DASH for ease of use.
 *
 * @param email The current user's email
 */
public void setUserEmail(String email)

/**
 *  Clears out local and cached user email data
 */
```

```
public void clearUserEmail()

/**
 * Used to set the push device token for the current app.
 * Set this with InstanceID token returned from the
InstanceIDService.
 * This is used to send DASH outbid notifications on the team's
behalf. Set each time token changes.
 * Token will be cached locally by DASH for ease of use.
 *
 * @param token The InstanceID token from the push service
 */
public void setPushToken(String token)

/**
 *  Clears out local and cached push tokens in DASH
 */
public void clearPushToken()

/**
 * Returns true if the intent extras passed in should be handled
by DASH.
 * If true, you should tell DASH to handle the push and present
the DASH fragment.
 * @param extras Intent extras on activity which handles push
 * @return If the intent extras passed in should be handled by
DASH
 */
public Boolean canHandlePushIntentExtras(Bundle extras)

/**
 * Tells DASH to handle the push. The next presentation of the
DASH fragment will handle the
 * notification. If DASH fragment is already presented, this
will reload the interface to handle
 * the notification. EX: If the DASH outbid push intent extras
are set here, the next
 * presentation will navigate directly to the respective auction
item.
 *
 * @param extras Intent extras on activity which handles push
 */
public void setPushIntentExtras(Bundle extras)

/**
 * Returns whether DASH currently has data as result of a push
```

```
 * (and subsequently DASH fragment should be presented)
 *
 * @return If DASH currently has push data
 */
public Boolean hasNotificationData()

/**
 * Clears out the pending notification data
 */
public void clearNotificationData()

/**
 * Returns a DASH fragment for display.
 * @return DASH fragment for display
 */
public DASHFragment dashFragment()
```

DASHFragment

A DASH fragment instance is the user interface for the DASH framework.

```
/**
 * Refreshes the current page. If startFromBeginning is true,
the interface is reloaded to the beginning state.
 *
 * @param startFromBeginning Whether the interface should be
reloaded to the beginning state.
 */
public void reloadInterface(boolean startFromBeginning)

/**
 * Used to set an event listener on the DASH fragment.
 * Currently broadcasts errors
 *
 * @param eventListener An event listener
 */
public void setEventListener(DASHFragmentEventListener
eventListener)
```

DASHFragmentEventListener

Event listener for the DASHFragment.

```
/**
 * Called each time an error occurs in the DASH fragment
 *
 * @param dashFragment The DASH fragment that had the error
 * @param errorCode The errorCode (passed through from
WebViewClient)
 * @param errorDescription A description of the error
 */
void onReceivedError(Fragment dashFragment, int errorCode,
String errorDescription);
```