Bazaarvoice iOS SDK Methods and Properties

The SDK is an iOS static library that provides an easy way to generate REST calls to the Bazaarvoice API. It also deserializes the response into a BVResponse object.

BVSettings

The BVSettings object is used to set the passkey, API version, customerName, data string, and format string (should always be JSON at this point) that is used throughout the API. By default, all the objects set the "settingsObject" to the singleton instance that can be obtained by calling [BVSettings instance] method. But there's no reason why this can't be set to another BVSettings object.

Consider a typical API call:
http://reviews.apitestcustomer.bazaarvoice.com/bvstaging/data/reviews.json?apiversion=5.1&passkey=kuy3zj9pr3n7i0wxajrzj04xo

Each of the properties set to a string that is used to construct the API call.

Methods

+ (BVSettings*) instance;

This is a class method to retrieve the singleton object that is used for all other objects.

Properties

@property (nonatomic, copy) NSString* passKey;

This is the passkey assigned to the customer. The default value is set to a passkey that works with the test server.

@property (nonatomic, copy) NSString* apiVersion;

The API version used. The default is set to 5.1

@property (nonatomic, copy) NSString* customerName;
The customer name used in the request. Default is set to reviews.apitestcustomer.

@property (nonatomic, copy) NSString* dataString;
dataString is the URL immediately after bazaarvoice.com/. The default is set to bvstaging/data used by the test server.

@property (nonatomic, copy) NSString* formatString;
At this point, this should always be left to the default setting "JSON". If XML parsing is implemented at a later date, this should be set to "XML".

BVBase

BVBase is the base class that actually constructs and executes the API calls. However, it is never called directly. Instead its subclasses override the necessary values and they are created and used by the developer instead. However, all the other classes are sub classed from BVBase, so it is important to understand this class.

Methods Called

- (void) startAsynchRequest;

This is the only method called by the instantiating class. It executes an asynchronous API request and will call one of 2 methods declared in the BVDelegate protocol.

- (NSString*) fragmentForKey:(NSString*)key
usingDictionary:(NSDictionary*)parametersDict;

This method generates the parameter values for each parameter stored in the dictionary. It generates the actual parameter string for each parameter that is either passed in the GET or POST request. Override this method in subclass if those classes generate parameter strings in a different way.

- (NSString*) parameterURL;

This method calls fragmentForKey to generate the URL to parameter string that is used in buildURLString

- (NSString*) buildURLString;

Builds the final URL string using the above methods to is sent to the BV API.

- (NSMutableURLRequest*) generateURLRequestWithString:(NSString*)string;

This method generates the actual URL request and sets any necessary HTTP headers. It is overridden in BVSubmission as those subclasses use a POST instead of a GET.

- (NSString*) contentType;
- (NSString*) displayType;

These 2 methods must always be overridden in sub classes to the appropriate values. They set the API method name before ".json" and the contentType in the BVResponse header.

Properties

@property (nonatomic, unsafe_unretained) id<BVDelegate> delegate;

Set to a delegate object that conforms to the BVDelegate protocol.

@property (nonatomic, strong) BVParameters* parameters;

The parameters property holds a reference to BVParameters or one if its subclasses that is used to generate the API request.

@property (nonatomic, readonly) NSString* rawURLRequest;

rawURLRequest is a read only property that contains the final URL request string to be sent to the server.

@property (nonatomic, strong) BVSettings* settingsObject;

settingsObject contains a reference to BVSettings storing the passkey, API and other common parameters. By default it is set to the singleton object [BVSettings instance].

@property (nonatomic, readonly) NSString* parameterURL;

This is a read only property declaration to access the final parameter URL generated for the buildURL method.

BVDelegate Protocol

BVDelegate is a protocol declaring the possible responses for each API request. Both are optional.

- (void) didReceiveResponse:(BVResponse*)response sender:(BVBase*)senderID;

If the request succeeds, the response is deserialized and sent in a BVResponse object. senderID is the object that made the API request.

- (void) didFailToReceiveResponse:(NSError*)err sender:(BVBase*)senderID;

If the request failed, a NSError object is returned along with the sender of the request.

BVParameters, BVParametersType

BVParametersType define multi type parameters such as filter, sort, and limit types. Consider the limit parameter type, which take the form of LIMIT_[TYPE] = value. There are 3 components to this parameter; the prefix (“LIMIT”), type (“Comments”, “Reviews”), and the value to be filtered. These values correspond to the 3 properties declared in BVParameters Type. It also has a dictionaryEntry method to collapse these properties into a single NSDictionary object.

BVParameters define parameters that are common to all API requests. It is a base class that is subclassed to add additional parameters specific to a particular API call.

It contains a single method, dictionaryOfValues that returns an NSDictionary key value list of the parameters set.

BVResponse

BVResponse is a list of properties that correspond to the possible responses sent from the server. If no value for that property was sent, it is set to nil or 0 depending on the type. It also contains 2 properties, rawURLRequest, and rawResponse that return the URL Request that was used and an NSDictionary of the JSON response.

Display Subclasses

BVDisplayReview, BVDisplayQuestions, BVDisplayAnswer, BVDisplayStories, BVDisplayComments, BVDisplayProfile, BVDisplayProducts, and BVDisplayCategories are all subclasses of BVBase. They override the displayType method used in each API request.

BVSubmission and Subclasses

BVSubmission is a base class subclassed by all the BVSubmission* classes. It overrides the parameters property with BVSubmissionParametersBase class, which is overrode again in its subclasses.

- (NSMutableURLRequest*) generateURLRequestWithString:(NSString*)string;

This method is overridden to generate an NSMutableURLRequest with a POST body instead of a GET that is used in the BVDisplay classes.

- (NSString*) buildURLString;

This method is overridden to not include parameters in the URL Request since Submissions are HTTP POSTs instead of GET's in Display requests.

BVSubmission subclasses override the displayType and contentType for those particular API's.

BVSubmissionPhoto

BVSubmissionPhoto is subclassed from BVSubmission and overrides generateURLRequestWithString: to generate a multi-form POST request. This is required to allow upload binary photo. The BVSubmissionParametersPhoto class contains a property to hold a UIImage. This UIImage object is converted to JPEG, then to an NSData that is included with a multi-part form.

BVSubmissionParametersBase and Subclasses

BVSubmissionParametersBase is a subclass of BVParameters. It is a subclass itself used for for other Submission types. BVSubmission* are its subclasses. They define the parameters used in each type of Submission request.