Secure Credit Systems

API Documentation

Secure Credit Systems

Table of Contents

Introduction

The following document provides general documentation applicable across all programming languages in which the SCS SDK is made available.

Application flow

SCS offers two application flows, depending on the integration choice. The first is score only and requires the least integration points. The second is full integration, which allows access to all of the features available on the SCS platform.

In either flow, the SDK must first be initialized by passing certain parameters, including the organization's authorization key and optionally, the phone number associated with the device. The authorization key is obtained through SCS and authorizes the organization's use of the SDK. If the mobile app collects the user's mobile phone number and it is provided to the SCS SDK, it can be used to identify the collected data and score and to assist in fraud prevention. If the mobile phone number is unavailable, pass an empty string.

Score Only

The score-only integration allows the SDK to collect metadata from the host mobile phone and return a score using SCS's proprietary technology. Implementation requires only a call to the Score function. The Score function will verify that the user has provided the necessary consent to the required permissions. Score will then collect the necessary metadata, interact with the SCS platform, and return a score and probability of default.

Full Integration

The Full Integration opens access to the full suite of features available on the SCS Platform. Full Integration requires that each consumer be registered as a User using the RegisterUser function. After registration, each User can log in with their email address and password. If login is successful, the SDK will receive a token that will automatically be sent with all subsequent calls.

After being logged in, a list of previously saved saved consents – if any -- can be retrieved for that User. The SDK can also save new consents for each User that each represent a date and time a particular type of consent was granted or denied.

To collect the User's data for assessment by the SCS platform, call the Score function, passing it the UserID received from the Login function. The Score function will verify that the user has provided the necessary consent to the required permissions. Score will then collect the necessary metadata, interact with the SCS platform, and return a score and probability of default.

Requirements

The SCS SDK requires the following Android permissions:

- android.permission.INTERNET
- android.permission.READ_EXTERNAL_STORAGE
- android.permission.READ_CONTACTS
- android.permission.READ_CALENDAR
- android.permission.GET_ACCOUNTS

- android.permission.BLUETOOTH
- android.permission.ACCESS_WIFI_STATE
- android.permission.USE_FINGERPRINT
- android.permission.QUERY_ALL_PACKAGES

SCSSDK

Initializes the SCSSDK with base parameters for the mobile app using the SDK and the particular mobile device.

Parameters:

- Authorization Key: Unique key provided to your organization to allow you to access the SCS Platform.
- API URL: Optional URL string specifying the location of the SCS API.

Returns:

- Initialized SCSSDK object.

Metadata Scoring Integration

Score

The Score function is used to collect alternative data from a consumer's mobile device and retrieve a credit risk score based on that metadata. The Score function verifies that the user has provided the necessary consent to the required permissions. These permissions include permission (1) to read contacts, (2) to read calendar, and (3) to read photos and media. If the correct permissions have not been granted, the score function will terminate. The Score function then collects the necessary metadata, interacts with the SCS platform, and returns a score and probability of default.

Parameters:

- Application Context: The current mobile application's context to enable permission checks and metadata gathering.
- At least one of the following unique identifiers:
  - Mobile number: an optional parameter used to uniquely identify the device and assist in fraud detection. If not collected, provide an empty string.
  - Email address: an optional parameter used to uniquely identify the device and assist in fraud detection. If not collected, provide an empty string.
  - Offer code: an optional parameter used to uniquely identify the device and assist in fraud detection. If not collected, provide an empty string.
  - User Id: User ID returned after calling "Register User"
-

Returns:

- ReferenceID (integer): The unique reference ID for this scoring collection.
- Score (integer): Relative risk score for the metadata collected on the particular device. E.g., 602.

- Probability (decimal): Decimal representing the relative probability that this user will default. Expressed with 4 decimal places. For example, 0.0532 reflects a relative 5.32% chance that the consumer will default.

Full Integration

Login

Function used to log in an existing user. Returns a User object that includes details about the user that were provided during registration.

Parameters:

- Email
- Password

Returns:

- Authenticated: Boolean value indicating whether the user has been logged in
- Token: JWT token that is required for all other SDK calls
- User Object:
  - UserId: a unique integer assigned to this user.
  - firstname: a string with the user's first name.
  - lastname: a string with the user's last name.
  - email: a string with the user's email address, e.g. user@example.com,
  - city: an optional string with the user's city.
  - country: an optional string with the user's country.
  - phone: a string containing the user's mobile phone number.
  - postalcode: a string containing the user's postal code.
  - password: a string containing a one-way hash of the user's password.

GetConsents

Used to retrieve a list of all consents – granted or denied – provided by a user.

Parameters:

- Userid: Integer user ID from the User object returned at login or registration.
- Token: JWT token returned from Login

Returns:

- Array of Consents:
  - userid: Integer user ID from the User object returned at login or registration.
  - consenttype: integer/enum representing the type of consent granted or denied. Enum values are:
    - 0 = location
    - 1 = SMS logs
    - 2 = Phone logs

    - 3 = Phone Info
    - 4 = Installed Apps
    - 5 = Contacts
    - 6 = Data Usage
    - 7 = App Usage
  - granted: The date and time at which the consent was granted. Null if permission is denied.
  - denied: The date and time at which the consent was denied. Null if permission is granted.

GetOfferData

Function to get the data related to the offer campaign to which the user is responding. The function takes an Offer code, which is a unique string associated with the campaign, and returns the particular campaign's initiation data.

Parameters:

- OfferCode: a string associated with the offer campaign to which the user is responding.
- Token: JWT token returned from Login

Returns:

- Initdata: Initiation data specific to the offer campaign.

GetUserData

Retrieves the data saved for this user

Parameters:

- UserId: Integer user ID from the User object returned at login or registration.
- StartDate: Start date for time interval for which data will be returned.
- EndDate: End date for time interval for which data will be returned.

Returns:

- Array of UserData:
  - userid: Integer user ID from the User object returned at login or registration.
  - Data: JSON string of data stored for user

RegisterUser

Function used to register a new user to the system.

Parameters:

- User Object:
  - firstname: a string with the user's first name.
  - lastname: a string with the user's last name.
  - email: a string with the user's email address, e.g. user@example.com,
  - city: an optional string with the user's city.
  - country: an optional string with the user's country.

  - phone: a string containing the user's mobile phone number.
  - postalcode: a string containing the user's postal code.
  - password: a string containing the user's password.

Returns:

- User Object:
  - Userid: a unique integer assigned to this user.
  - firstname: a string with the user's first name.
  - lastname: a string with the user's last name.
  - email: a string with the user's email address, e.g. user@example.com,
  - city: an optional string with the user's city.
  - country: an optional string with the user's country.
  - phone: a string containing the user's mobile phone number.
  - postalcode: a string containing the user's postal code.
  - password: the user's chosen password.

SaveConsent

Used to save a specific consent for a particular user.

Parameters:

- userid: Integer user ID from the User object returned at login or registration.
- consenttype: integer/enum representing the type of consent granted or denied. Enum values are:
  - 0 = location
  - 1 = SMS logs
  - 2 = Phone logs
  - 3 = Phone Info
  - 4 = Installed Apps
  - 5 = Contacts
  - 6 = Data Usage
  - 7 = App Usage
- granted: The date and time at which the consent was granted. Set to null if permission is denied.
- denied: The date and time at which the consent was denied. Set to null if permission is granted.
- Token: JWT token returned from Login

Returns:

- consentid: Unique ID of a particular consent.
- userid: Integer user ID from the User object returned at login or registration.
- consenttype: integer/enum representing the type of consent granted or denied. Enum values are:
  - 0 = location
  - 1 = SMS logs
  - 2 = Phone logs

  - 3 = Phone Info
  - 4 = Installed Apps
  - 5 = Contacts
  - 6 = Data Usage
  - 7 = App Usage
- granted: The date and time at which the consent was granted. Set to null if permission is denied.
- denied: The date and time at which the consent was denied. Set to null if permission is granted.

SaveData

Submit any data specific to a user in JSON format.

Parameters

- UserId: Integer user ID from the User object returned at login or registration.
- Data: JSON string of user data

Returns

- Success: Boolean whether data was saved correctly, true if success, false if not.
- DataId: Integer uniquely identifying data saved.

VerifyConsent

Verifies that an existing consent is still valid.

Parameters

- Consent ID: Unique ID of a particular consent.
- UserID: Integer user ID from the User object returned at login or registration.
- consenttype: integer/enum representing the type of consent granted or denied. Enum values are:
  - 0 = location
  - 1 = SMS logs
  - 2 = Phone logs
  - 3 = Phone Info
  - 4 = Installed Apps
  - 5 = Contacts
  - 6 = Data Usage
  - 7 = App Usage

Returns

- consentid: unique ID of a particular consent.
- userid: Integer user ID from the User object returned at login or registration.
- consenttype: integer/enum representing the type of consent granted or denied. Enum values are:
  - 0 = location

  - 1 = SMS logs
  - 2 = Phone logs
  - 3 = Phone Info
  - 4 = Installed Apps
  - 5 = Contacts
  - 6 = Data Usage
  - 7 = App Usage
- granted: The date and time at which the consent was granted. Set to null if permission is denied.
- denied: The date and time at which the consent was denied. Set to null if permission is granted.

Sample Code

```
//add the libraries to your gradle dependencies
implementation files('libs/scorelib.aar')
implementation files('libs/scssdk.aar')

//import the SCS SDK
import com.scs.sdk.SCSSDK


//create an array of permissions to verify
val arrPerms = arrayOf(
Manifest.permission.WRITE_EXTERNAL_STORAGE,
Manifest.permission.READ_EXTERNAL_STORAGE,
Manifest.permission.READ_CONTACTS,
Manifest.permission.READ_CALENDAR,
Manifest.permission.GET_ACCOUNTS,
Manifest.permission.BLUETOOTH,
Manifest.permission.ACCESS_WIFI_STATE,
Manifest.permission.ACCESS_NETWORK_STATE,
Manifest.permission.USE_FINGERPRINT,
Manifest.permission.QUERY_ALL_PACKAGES
)
//Verify that the user has given the necessary permissions – if they haven’t, request the permission.
ActivityCompat.requestPermissions(this@MainActivity, permission, requestCode)
//collect the user’s email address as a unique identifier for the user
val email = “test2002@test.com”
//retrieve the authorization key from settings or wherever you securely store it
val authKey = PROVIDED_BY_SCS
//initialize the SCSSDK
var scs = SCSSDK(authKey)
//call the score function passing
val score = scs.score(applicationContext, email=email).toString()
```

Sample Score Result

```
{
          "scoreid": 32486,
          "scorevalue": 655,
          "reference": "163659741542",
          "probability": 0.0148,
          "scorecode": 1,
          "device": {
            "os": "Android",
            "brand": "Android",
            "model": "Android SDK built for x86",
            "browser": null,
            "identifier": "0fa59f2bd865050584ac39dcbcc3cd9b",
            "associatedDatasetsCount": 5
          },
          "permissions": [
            { "name": "android.permission.READ_CONTACTS", "value": 1 },
            { "name": "android.permission.READ_CALENDAR", "value": 1 },
            { "name": "android.permission.READ_EXTERNAL_STORAGE", "value": 1 },
            { "name": "android.permission.GET_ACCOUNTS", "value": 1 },
            { "name": "android.permission.BLUETOOTH", "value": 1 },
            { "name": "android.permission.ACCESS_WIFI_STATE", "value": 1 },
            { "name": "android.permission.USE_FINGERPRINT", "value": 1 },
            { "name": "android.permission.QUERY_ALL_PACKAGES", "value": 1 }
          ],
          "collectionstartdate": "2020-11-11T02:23:41.127Z",
          "collectionenddate": "2020-11-11T02:23:45.817Z",
          "rawdatasetsize": null,
          "isrepeat": true,
          "offercode": "",
          "declinethreshold": 500,
          "prequalthreshold": 550,
          "userid": 10064,
          "campaignid": "-1"
        }
```