Technical Manual
Login for iOS & Android

Updated: February 2021

DocCheck

Index

1 The DocCheck App Login

With the DocCheck App Login you can conveniently conform your medical applications to your needs while making them healthcare compliant (HCC).

The DocCheck App Login can be used with either iOS or Android for the purpose of authenticating a user's professional affiliation - you can thereby direct access of different profession groups to your app via our CRM facility DocCheckCReaM (https://crm.doccheck.com/com/)

Please note: You will require Mobile Login Framework for integration of the DocCheck Login. Please contact us for assistance with this.

The DocCheck App Login is offered in three variants:

1.1 App Login Basic – The classic among apps

- Integration of login via the framework supplied by us
- Professional group checking and filtering via DocCheck CReaM

1. to 3. app: €700*

4. to 6. app: €630*

7. to 9. app: €560*

+ €250 setup fee

* Prices apply per calendar year and app.

1.2 App Login Economy – Sort through your users

- All elements of the Basic App Login
- Using the unique key you can recognise your users anonymously at their next visit
- Easy access for your company's employees, while at the same time excluding competitors

1. to 3. app: €1,500*

4. to 6. app: €1,350*

7. to 9. app: €1,200*

+ €1,200 setup fee

1.3 App Login Business – Get to know your users

- All elements of the Basic and Economy App Logins
- Personal data submitted following consent from the user, so that you can construct your own user database
- Easy access for your company's employees, while at the same time excluding competitors

1. to 3. app: €4,500*

4. to 6. app: €4,050*

7. to 9. app: €3,600*

+ €4,200 setup fee

* Prices apply per calendar year and app.

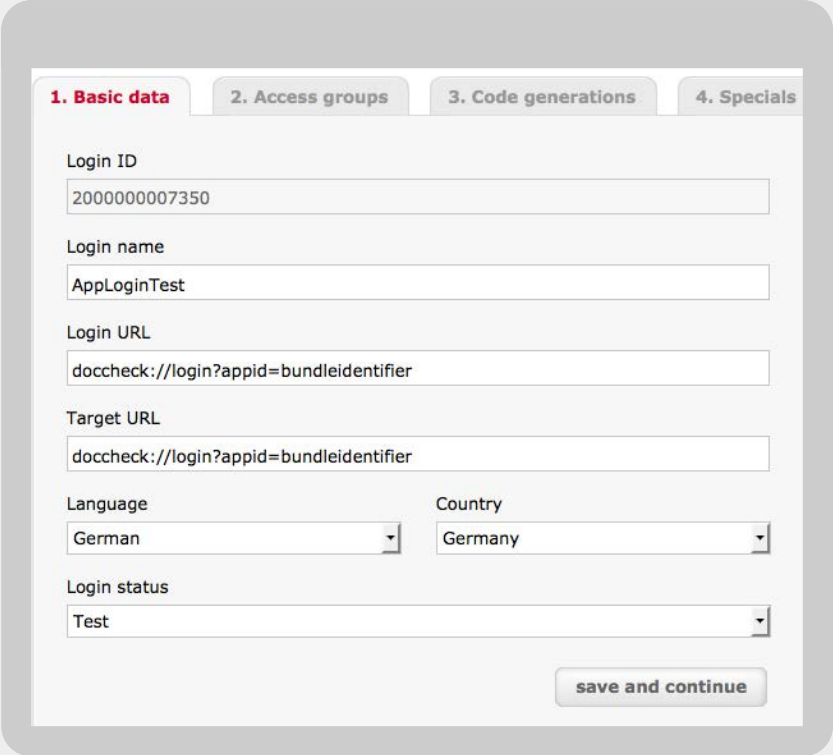
2 Implementing a DocCheck Login

Please note: You will require the matching iOS or Android framework when implementing the DocCheck Login into your app.
Please contact us for assistance with this.

In order to ensure smooth integration and functioning always adhere to using the respective framework.

Before you can implement the DocCheck Login in your app, you will need to set up a new login in CReaM (http://crm.doccheck.com/com/).
For detailed instructions please read the technical handbook under http://biz.doccheck.com/com/services/passwordprotection/ (chapter 2.2.3).

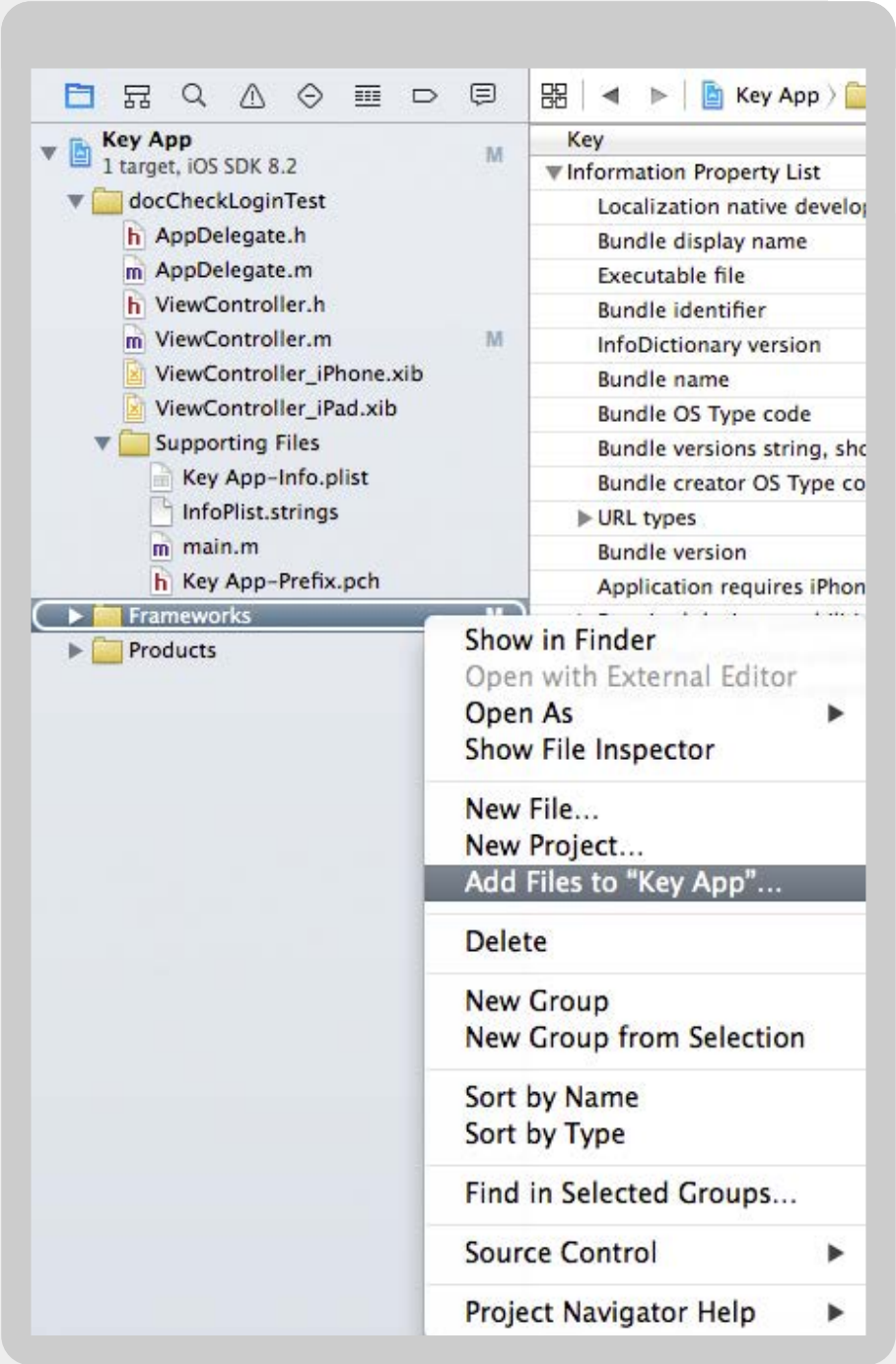
When a new login has been created, please add the bundle identifier (for iOS) or the package name (for Android) in the destination URL. This process in general runs as follows: **topleveldomain.companyname.appname**



2.1 Implementation within a native iOS app

We recommend that the latest Xcode version of the framework always be used with an iOS app.

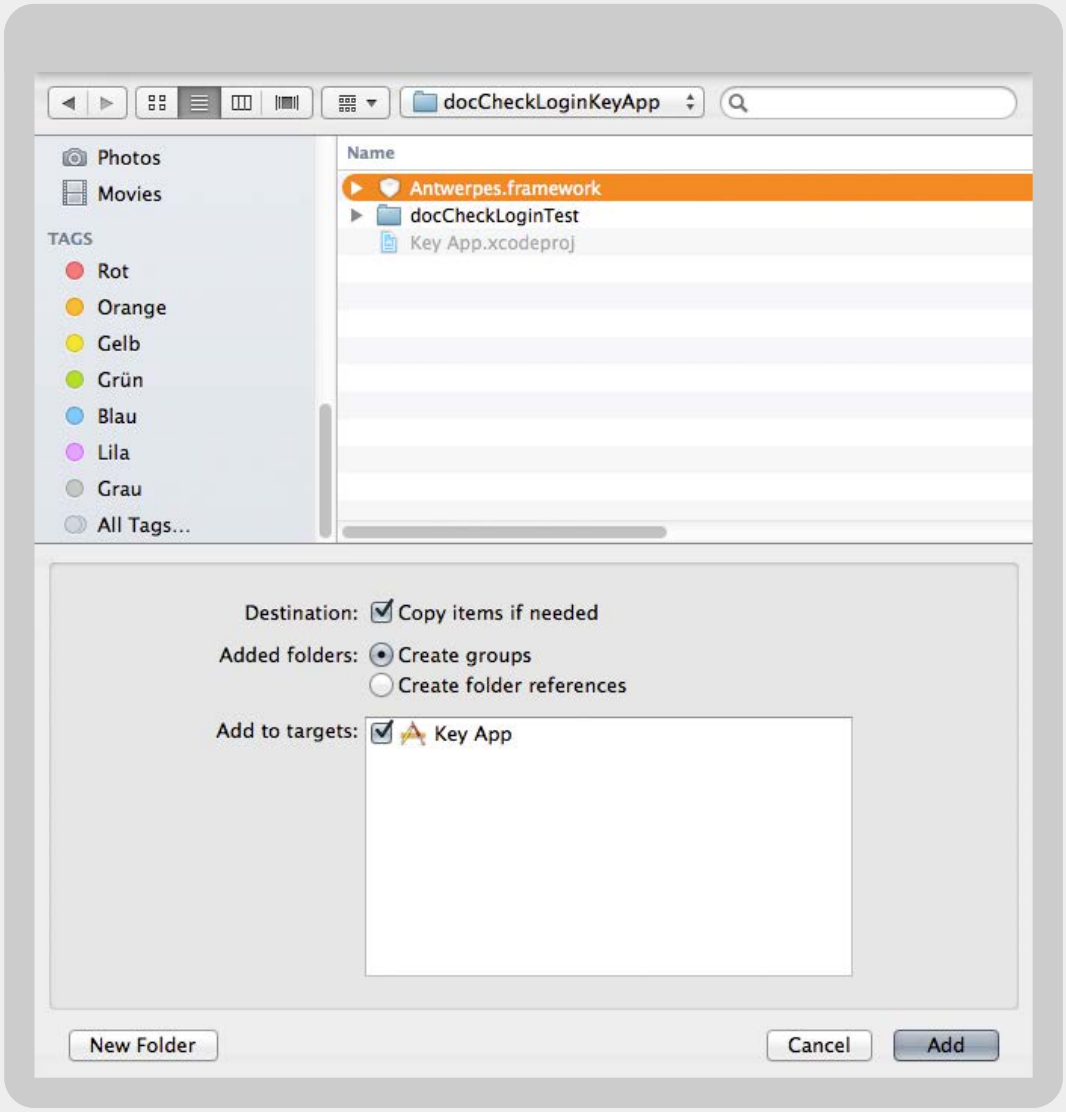
2.1.1 Add Antwerpes.framework



In order to be able to use the DocCheck Login within your app, first you will need to add the **Antwerpes.framework** to your project.

To do this, click using the right mouse button **"Frameworks" → "Add" → "Existing Files...".**

Navigate to the directory in which you have unpacked the **Antwerpes.framework** folder. Insert a tick in "**Copy items if needed**". Then select the framework and click on "**Add**".



2.1.2 Modify controller

First, create a new login in your CreaM account (see point 2 or the technical manual). During configuration, you will receive the login ID. The app ID is then to be entered in the start and destination URL fields.

Import the **Antwerpes.framework** into the .h file of a class (e.g. a view controller) and add the **DocCheckLoginDelegate** protocol.

An example of the class **DocCheckLoginTestViewController.h**:

```
#import <Antwerpes/APDocCheckLogin.h>
@interface
DocCheckLoginTestViewController :
UIViewController<DocCheckLoginDelegate> {
  …
}
…
@end
```

In order for the DocCheck login form to be shown, the following code must be used (for example in an **IBAction**):

```
- (IBAction)
startLogin {
  APDocCheckLogin*dcl = [APDocCheckLogin sharedInstance];
  dcl.loginId= @"1234567890123"; // Your DocCheck Login ID
  dcl.delegate=self;
  // Available languages: de, com, es, fr, nl, it
  dcl.language = @"de";
  // Adds the DocCheck Login to your view
  self.view addSubview:dcl.view];
}
```

In addition, the following **DocCheckLoginDelegates** methods need to be implemented:

```
- (void)docCheckLoginSuccessful;
```

→ Invoked as soon as the login is successfully concluded. Now the program can be run as normal. The DocCheck Login view will be removed on its own.

```
- (void)docCheckLoginFailedWithError:(NSError * ) error;
```

→ Invoked when the login fails because of a lost internet connection. The users should be informed about the error, and any error should be dealt with appropriately.

The following DocCheckLoginDelegates methods can be optionally implemented:

```
- (void) docCheckLoginCanceled;
```

→ Invoked when the login is cancelled by the user manually by using via the "back" arrow.

```
- (void) docCheckLoginReceivedUserInfo: (NSDictionary * )userInfo;
```

→ Invoked after the login has been completed successfully.
userInfo contains detailed information on the user profile where the user has previously consented to the use of the relevant data in his/her profile settings.

2.1.3 Configuration for iOS9+

Due to new restrictions in iOS9, it's necessary to define an appropriate exception, so that the connection to DocCheck can be established. Therefore ensure that the following entries are present in the **Info.plist** file of your iOS project:

```
<key>NSAppTransportSecurity</key>
<dict>
  <key>NSExceptionDomains</key>
  <dict>
    <key>dccdn.de</key>
    <dict>
      <key>NSExceptionAllowsInsecureHTTPLoads</key>
      <true/>
      <key>NSExceptionMinimumTLSVersion</key>
      <string>TLSv1.0</string>
      <key>NSExceptionRequiresForwardSecrecy</key>
      <false/>
      <key>NSIncludesSubdomains</key>
      <true/>
      <key>NSRequiresCertificateTransparency</key>
      <false/>
      <key>NSThirdPartyExceptionAllowsInsecureHTTPLoads</key>
      <true/>
      <key>NSThirdPartyExceptionMinimumTLSVersion</key>
      <string>TLSv1.0</string>
      <key>NSThirdPartyExceptionRequiresForwardSecrecy</key>
      <false/>
    </dict>
    <key>doccheck.com</key>
    <dict>
      <key>NSExceptionAllowsInsecureHTTPLoads</key>
      <true/>
      <key>NSExceptionMinimumTLSVersion</key>
      <string>TLSv1.0</string>
      <key>NSExceptionRequiresForwardSecrecy</key>
      <false/>
      <key>NSIncludesSubdomains</key>
      <true/>
      <key>NSRequiresCertificateTransparency</key>
      <false/>
      <key>NSThirdPartyExceptionAllowsInsecureHTTPLoads</key>
      <true/>
      <key>NSThirdPartyExceptionMinimumTLSVersion</key>
      <string>TLSv1.0</string>
      <key>NSThirdPartyExceptionRequiresForwardSecrecy</key>
      <false/>
    </dict>
  </dict>
</dict>
```

2.1.4 Customise login interface

Please note: So that we can also modify the login interface of your app for you, the associated special (as specified by the licence model) needs to be activated by DocCheck. For this reason please let us know the ID of your DocCheck Login and your AppID (bundle identifier).

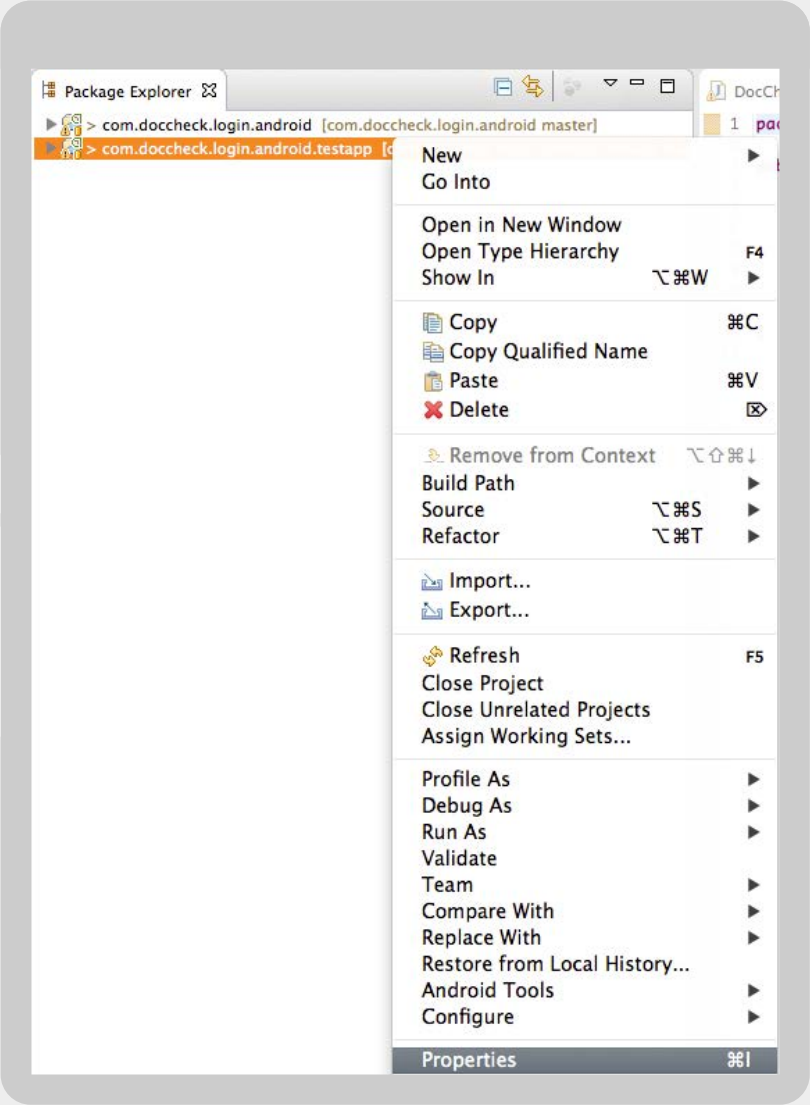
2.2 Implementation within a native Android app

When using the DocCheck Login the calling app has to integrate the library project "DocCheckLogin". The Android project DocCheckLoginTest is present as a supplement to DocCheckLogin Library as an exemplar, so as to make the integration and invoking of the login more clearly understandable.

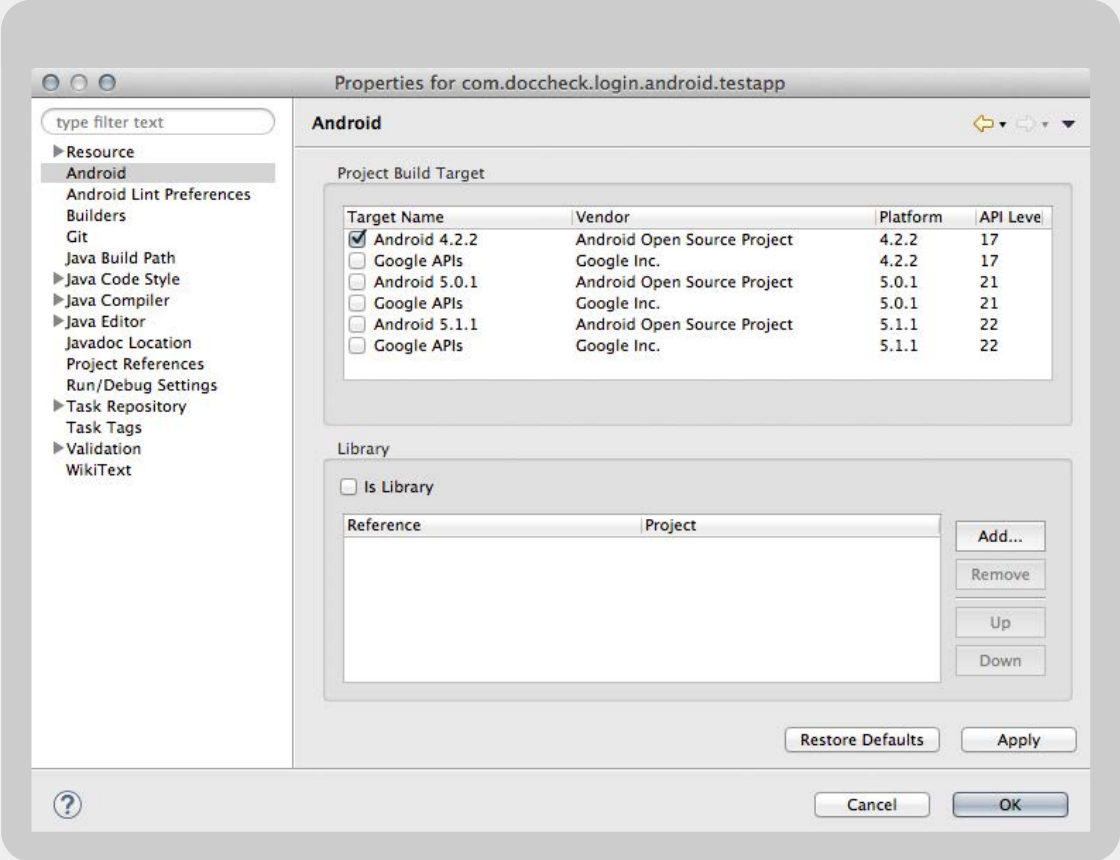
2.2.1 Add Library Project

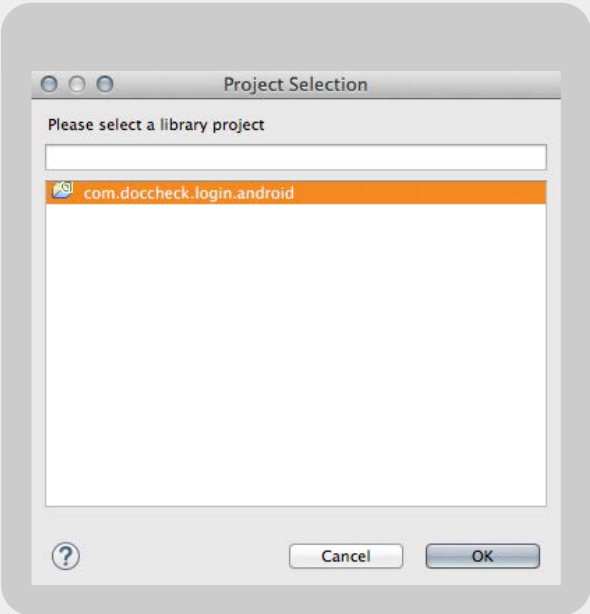
In order to be able to use the DocCheck Login within your app, you must first add the **Library Project** to your project.

Right-click in the project explorer on your project and select **Properties**.



In the next window, select the submenu **Android**. Click in the section library on the "**Add...**" -button.





Select DocCheck Login Library project from the list. Should the project not appear in the list, make sure that the project has been placed in your workspace. Carry out your selection.

2.2.2 Integrating DocCheck Login Application

Introductory remarks:

- The package identifier (e.g. com.example.app) used for for the application must match that deposited with DocCheck
- Library project includes dcview.xml, this name is therefore reserved for the DocCheck Login and may not be used in your own project
- The following entries must be additionally supplied in the Android manifest file in order to use DocCheck Login:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<activity android:name="com.doccheck.login.android.DocCheckLogin" android:configChanges="orienta-
tion|keyboard|keyboardHidden"/>
```

The following entries as well as the following code snippets can also be taken from the accompanying sample project.

Definition of the interface:

```
package com.doccheck.login.android.test;
public interface DCTestActivityInterface {
  public static final String loginId = "Your login id";
  public static final String templateName = "s_mobile";
  public static final int SHOW_DC_LOGIN = 1;
  public static final String CONST_LOGINID = "loginId";
  public static final String CONST_TEMPLATENAME = "templateName";
  public static final String CONST_LANGUAGE = "lang";
}
```

Invoking the Library Project from within the app:

```
Intent i = new Intent(DocCheckLogin2TestActivity.this, DocCheckLogin.class);
i.putExtra(CONST_LOGINID, loginId);
i.putExtra(CONST_TEMPLATENAME, templateName);
i.putExtra(CONST_COUNTRYCODE, selectedLang);
startActivityForResult(i, SHOW_DC_LOGIN);
```

DocCheckLogin2TestActivity.this designates the invoking activity and must be adapted accordingly to the relevant project. DocCheckLogin.class is the login activity of the library.

The country code is selected in the sample project via a spinner, it may of course however also be given in other ways. Valid language codes include **de, com, es, fr, nl, it**.

In order to evaluate the result of the login, the method onActivityResult must be implemented. The login result is read via data.getBooleanExtra ("LOGIN_RESULT", false). Additional parameters (e.g. title, first name, surname, etc.) are read via data.getSerializableExtra ("URLPARAMS"). Access to individual parameters is implemented as follows: String anrede = paramsMap.get("dc_anrede").get(0);

One of the additional parameters needs to be checked against zero, since a valid login does not have to necessarily involve a parameter. This is the case where the page is visited anonymously (see sample project, class DocCheckLogin2Activity.java, method onActivityResult).

The following parameters can be read (the requisite here is the Business License, see 1.3):

- Appid
- dc_anrede, dc_gender, dc_titel, dc_vorname, dc_name
- dc_strasse, dc_plz, dc_ort, dc_land
- dc_beruf, dc_fachgebiet, dc_email

2.3 Implementation within a Hybrid app

2.3.1 Setup

The basis will be hybrid app development, in this case via Cordova.

Please note: For the implementation of a hybrid app the respective plugin will be required. Please contact us about this.

1. Unzip the plugin archive in a location of your choice.
2. The adding process does not differ from that of other CordovaPlugins. It should however be ensured that the correct local path gets specified. Switch from the command line in your project directory and carry out the following command:

   cordova plugin add /path/to/local/plugincopy

3. After installation, dclogin.js must be referenced in the index.html:

   <script type="text/javascript" src="js/dclogin.js"></script>

2.3.2 Configuration for iOS9+

Due to new restrictions in iOS9, it's necessary to define an appropriate exception, so that the connection to DocCheck can be established. Therefore ensure that the following entries are present in the **Info.plist** file of your iOS project:

```
<key>NSAppTransportSecurity</key>
<dict>
  <key>NSExceptionDomains</key>
  <dict>
    <key>dccdn.de</key>
    <dict>
      <key>NSExceptionAllowsInsecureHTTPLoads</key>
      <true/>
      <key>NSExceptionMinimumTLSVersion</key>
      <string>TLSv1.0</string>
      <key>NSExceptionRequiresForwardSecrecy</key>
      <false/>
      <key>NSIncludesSubdomains</key>
      <true/>
      <key>NSRequiresCertificateTransparency</key>
      <false/>
      <key>NSThirdPartyExceptionAllowsInsecureHTTPLoads</key>
      <true/>
      <key>NSThirdPartyExceptionMinimumTLSVersion</key>
      <string>TLSv1.0</string>
      <key>NSThirdPartyExceptionRequiresForwardSecrecy</key>
      <false/>
    </dict>
    <key>doccheck.com</key>
    <dict>
      <key>NSExceptionAllowsInsecureHTTPLoads</key>
      <true/>
      <key>NSExceptionMinimumTLSVersion</key>
      <string>TLSv1.0</string>
      <key>NSExceptionRequiresForwardSecrecy</key>
      <false/>
      <key>NSIncludesSubdomains</key>
      <true/>
      <key>NSRequiresCertificateTransparency</key>
      <false/>
      <key>NSThirdPartyExceptionAllowsInsecureHTTPLoads</key>
      <true/>
      <key>NSThirdPartyExceptionMinimumTLSVersion</key>
      <string>TLSv1.0</string>
      <key>NSThirdPartyExceptionRequiresForwardSecrecy</key>
      <false/>
    </dict>
  </dict>
</dict>
```

2.3.3 Invoking the DocCheck login form

After that, the login may be invoked. The following is an example of usage:

```
window.dclogin("1234567890000", "de", function(result) {
  if(result == 1)
    alert("Logged in!");
  else if(result == 0)
    alert("Not logged in!");
});
```

The parameters appear as follows:

dclogin(loginId, countryCode, callback)

loginId – The Login ID provided by DocCheck

countryCode – "de", "com", "es", "fr", "nl" or "it" (indicated at least in the native implementation as valid values).

callback – A function in which the client can accommodate his/her own logic. The parameter (as indicated in the above example result) gives information here about whether the login was successful. A value of 1 represents a successful login, 0 represents failure.

2.3.4 Sending a request from an own form

Should a separate form have been created for the DocCheck Login, a function is available to deal with the necessary login request.

```
window.dcloginrequest("1234567890000", function(result) {
  if(result)
    alert("Logged in!");
  else
    alert("Not logged in!");
});
```

The parameters appear as follows:

dcloginrequest(loginId, callback <, usernameId, passwordId>)

loginId – The Login ID provided by DocCheck

callback – A resultparameter of Boolean type is conveyed to the callback. If the result is true, the login attempt was successful, false means the attempt failed.

usernameId, passwordId – If these parameters are not conveyed, the plugin expects standard form fields with the IDs dc_username and dc_password.

Should the input fields in the form not have been filled with other IDs then the appropriate IDs need to be given here, so that the values from plugin can be read and evaluated. Instead of giving IDs, names can be used in order to identify the input fields.

2.4 Development and testing environment

Xcode 6.2
Eclipse Luna

2.4.1 iOS (Xcode) – Devices/Versions

iPhone 4 (iOS 7)
iPhone 5 (iOS 8)
iPhone 6 (iOS 8)
iPhone 6 Plus (iOS 8)
iPad Air (iOS 8)
iPad 4 (iOS 8)

2.4.2 Android – Devices/Versions

HTC Desire (Android 2.3)
Samsung Galaxy S4 Mini (Android 4.4)
HTC One (Android 5.0)
Nexus 9 (Android 5.0)

3 Your contact

Lara Albrecht
Product Managerin

+49 221 92053-595
industry@doccheck.com

David Steiger
Junior Product Manager

+49 221 92053-595
industry@doccheck.com

DocCheck Community GmbH
Vogelsanger Str. 66, 50823 Cologne, Germany
www.doccheck.com/com/

DocCheck