BlackBerry Dynamics Application Developer Guide

Bypass Unlock

Contents

Introduction

Bypass Unlock is a feature for BlackBerry Dynamics applications. It allows the scope of the BlackBerry Dynamics idle lock to be configured. Part of the application user interface can then remain accessible after the idle time out has expired.

By default, the whole of a BlackBerry Dynamics application's user interface is within the scope of the idle lock. After the idle time out has expired, the BlackBerry Dynamics unlock screen is superimposed on the application user interface. This doesn't apply to screens that are configured for Bypass Unlock. Those screens are immediately accessible, without end user authentication. Note that those screens are therefore not protected by the BlackBerry Dynamics idle lock.

Some general use cases of Bypass Unlock are:

- Notifications that require a swift response by the end user, such as an inbound call in a Voice Over IP (VoIP) application.
- Constant display of less sensitive application data.
- Immediate storage of new data, such as a note or photograph, to the secure store.

Use of the Bypass Unlock feature is an application privilege. It must be requested from BlackBerry, who will require that the application implements the feature according to the rules in this document.

Availability

The Bypass Unlock feature is supported by the following software development kit (SDK) products:

- Good Dynamics SDK for Android version 2.3 and later.
- Good Dynamics SDK for iOS version 2.3 and later.
- BlackBerry Dynamics SDK for Android version 3 and later.
- BlackBerry Dynamics SDK for iOS version 3 and later.

The name of the product was changed to BlackBerry Dynamics in version 3. There is no Good Dynamics version 3, nor BlackBerry Dynamics earlier than version 3.

Background

For an introduction to BlackBerry Dynamics, see the Platform Overview for Administrators and Developers. It is published here:
http://help.blackberry.com/en/good-control-good-proxy/current/

Bypass Unlock in the BlackBerry Dynamics authentication cycle

Bypass Unlock applies at a particular point in the authentication cycle.

A BlackBerry Dynamics application goes through the following authentication states at start-up.

Not authenticated. The initial state. The application will be in this state until the end user has entered their security password for the first time after start-up. Note that the end user could supply a different authentication secret, instead of a password, if a Trusted Authenticator is in use.

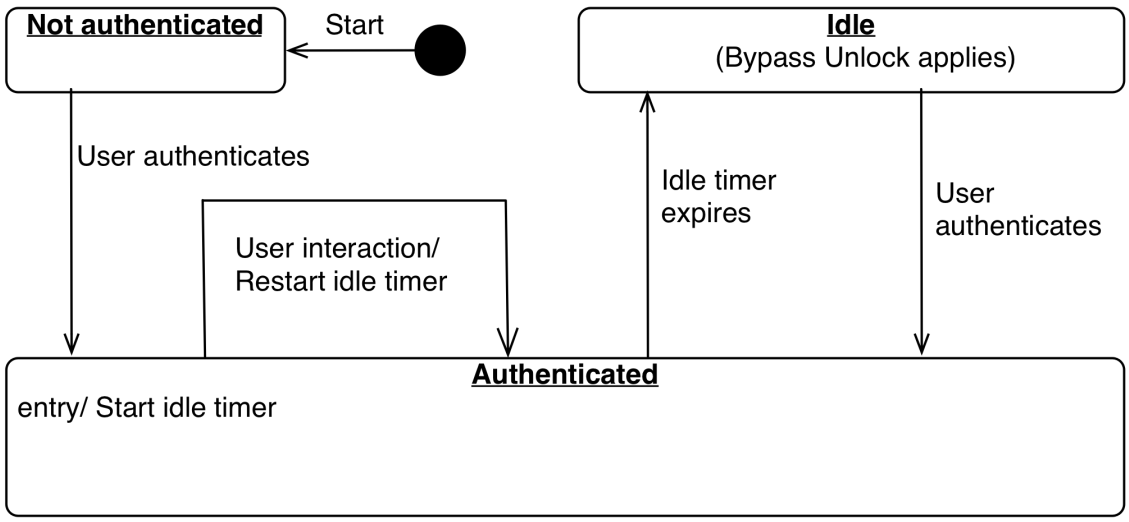
Authenticated. The state immediately after the user has supplied their password, or other authentication secret. Data on the device is protected with an encryption key that is derived from the secret. After the secret has been supplied, the key is derived and the runtime can access its management data, and any application data.

Idle. The state entered when the user hasn't interacted with the application for the idle time out. The duration of the idle time out is set in the management console, as a security policy setting. The runtime locks the application user interface when the Idle state is entered.

Bypass Unlock applies in the Idle state only.

Diagram

The above states are illustrated in the following diagram.



Name	st06 Authentication and Idle Lock
Documentation	UML State Transition diagram for initial authentication and idle lock

Notes on authentication state

If the device is switched off and on, and then the application is started again, it will be back in the initial state: Not authenticated. The same is true when the application is started after having been unloaded from memory. The application could be unloaded by the device operating system to release resources, or if it crashes, or if the user chooses to terminate it.

Bypass Unlock doesn't apply in the Not authenticated state. For example, it doesn't apply in the case that the application is unloaded from memory and then restarted. Showing any screen when not authenticated will cause the unlock screen to be displayed.

Division of the user interface

For the purposes of Bypass Unlock, the user interface of a BlackBerry Dynamics mobile application is divided into two sets of screens.

Bypass. Screens that could be displayed even if the application user interface is locked by BlackBerry Dynamics, in the Idle state.

Protected. All other screens, which will never be displayed when the application user interface is locked.

Bypass Unlock Policy

Bypass Unlock can be allowed or disallowed by enterprise policy. This is implemented using the BlackBerry Dynamics Application Policies feature. (Application Policies are sometimes known as Application-Specific Policies.)

The end user of a BlackBerry Dynamics application is always associated with an enterprise, by activation. The enterprise can set a number of policies that control use of the application by the end user, including allowing or disallowing Bypass Unlock.

If the enterprise disallows Bypass Unlock then Bypass screen designations are ignored and every screen is treated as a Protected screen. The user interface will then be locked when in the Idle state, regardless of which application screen is displayed.

Bypass Unlock Registration

Access to Bypass Unlock is restricted and must be requested from BlackBerry. Applications that are granted access to Bypass Unlock are registered by BlackBerry, and each issued a unique registration message. The message must be embedded in the application declaration, at build-time.

Summary of Bypass Unlock conditions

The conditions for display of a screen if the application is in the Idle locked state are:

- The particular screen has been marked for Bypass Unlock in the application declaration.
- The end user has authenticated at least once since the application started.
- Bypass Unlock is allowed by enterprise policy.
- A valid Bypass Unlock registration message is embedded in the application.

Implementation Essentials

To implement Bypass Unlock in your application, you would typically do the following:

- Declare and implement your Bypass screens.
- Create a Bypass Unlock policy definition for your application.
- Implement tracking of the Idle lock state.

Each of these items is detailed in a sub-section, see below.

Start with a copy of a Bypass Unlock sample application

The software development kits for mobile platforms that support Bypass Unlock each come with a sample application. A copy of the Bypass Unlock sample application can be used as a starting point for your own implementation.

Access to the Bypass Unlock feature is restricted and requires registration of the application identifier, and other details. The sample applications' identifiers are pre-registered, and a signed registration message is embedded in their application declarations.

If there isn't a valid Bypass Unlock registration message embedded in the application declaration, every screen will be treated as a Protected screen. The user interface will then be locked when in the Idle state, regardless of which application screen is displayed.

Register for access to the feature when complete

When your implementation is finished, or nearly finished, request access to the Bypass Unlock feature for your application.

The procedure for registration is documented in a separate section, below.

When registration is complete, stop using the identifiers of the Bypass Unlock sample applications and use the official identifiers of your application instead. The application will then utilise Bypass Unlock and can be released to production or beta.

Declare and implement Bypass screens

Declare which screens are Bypass in your application. You might also need to implement new screens. The possible need for new screens is discussed in the Discussion section, below.

Declarations and user interface implementations are platform specific.

- In an Android application, a Bypass screen will be an Activity.

  Add a meta-data tag to the declaration of each Bypass Screen in the `AndroidManifest.xml` file, as shown in the following snippet.

```
<activity
    android:name=".InCallActivity"
    android:configChanges="orientation|keyboardHidden|screenSize">
    <!-- Declare this Activity as a Bypass Screen. -->
    <meta-data android:name="com.good.gd.bypassunlock"
               android:value="true"/>
</activity>
```

  Any mechanisms for launching an Activity can be used to display a Bypass screen. For example, the following snippet shows how to launch as a child Activity.

```
Intent intent = new Intent(ParentActivity.this, BypassActivityName.class);
intent.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
startActivity(intent);
```

- In an iOS application, a Bypass screen will be a View Controller.

  List the View Controller class names for Bypass Screens in a property in the Info.plist file, as shown in the following snippet.

```
<key>GDBypassUnlockViewControllers</key>
<array>
    <string>BPIncallViewController</string>
</array>
```

  A View Controller that can be pushed onto the user interface navigation stack can be a Bypass screen. In programming terms, this means a `UIViewController` instance that could be passed successfully to the `UINavigationController pushViewController` function.

  Any of the following mechanisms can be used to display Bypass screens.

  - Functions in the UIKit programming interface, such as:
    `UIViewController presentViewController:animated:completion:`
  - Segue with modal presentation, configured in a storyboard.
  - Programmatic segue, such as:
    `UIViewController performSegueWithIdentifier:sender:`

Create a Bypass Unlock policy definition

Define a custom application policy that will control use of Bypass Unlock by end users.

- Create an Application Policies definition file, if your application doesn't already have one.

  For an introduction to Application Policies see the technical brief, which is published on the application developer portal here: https://community.good.com/docs/DOC-1543

- Add a setting for the Bypass Unlock policy.

  The following snippet shows an example setting definition.

```
<setting name="GD_SDK_Security_AllowBypassUnlock" >
    <checkbox>
        <key>GD_SDK_Security_AllowBypassUnlock</key>
        <label
            >Allow parts of the user interface to be displayed when
            idle lock is in place.</label>
        <value>false</value>
    </checkbox>
</setting>
```

  In the above snippet, the name of the setting is GD_SDK_Security_AllowBypassUnlock.

- Include the setting in the structural part of the Application Policies file.

  The following snippet shows an example of the structural part of the definition.

```
<pview>
    <pview type="tabbed" key="BlackBerryDynamicsFeatures">
        <title>BlackBerry Dynamics Features</title>
        <pe ref="GD_SDK_Security_AllowBypassUnlock" />
        <desc> - Incoming Call Screen</desc>
        <desc> - In-Call Screen </desc>
    </pview>
</pview>
```

  Note that a list of Bypass screens is included. The list will appear in the policy set editor in the management console.

- Check the policy setting at run time.

  - In an Android application, call the `GDAndroid getApplicationPolicy` or `getApplicationPolicyString` method to get the settings as a collection or as a JavaScript Object Notation (JSON) string, respectively.
  - In an iOS application, call the `GDiOS getApplicationPolicy` or `getApplicationPolicyString` function to get the settings as a collection or JSON string, respectively.

  The name of the item in the collection or JSON string will be the same as in the declaration. In the above example, it is `GD_SDK_Security_AllowBypassUnlock`.

Declare the policy setting name

The name of the Bypass Unlock policy setting must be declared, in the application declaration file. Application declaration files are platform specific.

- In an Android application, add the declaration to the `settings.json` file.

  Add a setting like the following:

  ```
  "GDBypassUnlockPolicySetting": "GD_SDK_Security_AllowBypassUnlock"
  ```

  The setting must be at the top level of the hierachy. This is the same level as the GDApplicationID and GDApplicationVersion settings.

- In an iOS application, add the declaration to the Info.plist file.

  Add a setting like the following, in the XML view:

  ```
  <key>GDBypassUnlockPolicySetting</key>
  <string>GD_SDK_Security_AllowBypassUnlock</string>
  ```

  In the property list view, the same declaration would appear like this:

Key	Type	Value
GDBypassUnlockPolicySetting	String	GD_SDK_Security_AllowBypassUnlock

  The setting must be at the top level of the hierachy. This is the same level as the GDApplicationID and GDApplicationVersion settings.

In the above examples, the declared policy setting name is GD_SDK_Security_AllowBypassUnlock. This is also the name in the examples in the above section, Create a Bypass Unlock policy definition. It is possible to use a custom name, in case GD_SDK_Security_AllowBypassUnlock is somehow unsuitable for your application. Replace GD_SDK_Security_AllowBypassUnlock with your custom name wherever it occurs in the application policy definition, and in the application declaration.

Implement tracking of the Idle lock

Track when the idle lock is in place. Doing so will enable the application to avoid attempting to display a Bypass-only screen if the application is idle locked and Bypass Unlock isn't allowed by policy.

- In an Android application, when the idle timer expires, the `GDAppEventListener` instance `onGDEvent` method will be despatched a `GDAppEvent` with the following characteristics:
  - The `getEventType()` accessor will return `GDAppEventNotAuthorized`.
  - The `getResultCode()` accessor will return `GDErrorIdleLockout`.
- In an iOS application, when the idle timer expires, the `GDiOSDelegate` instance `handleEvent:` function will be despatched a `GDAppEvent` with the following property values:
  - `type GDAppEventNotAuthorized`.
  - `code GDErrorIdleLockout`.

If the application attempts to open a Bypass screen when Bypass Unlock is disallowed, the Bypass designation will be ignored and the unlock screen will be superimposed.

Registration

Access to Bypass Unlock is restricted and must be requested from BlackBerry. Applications that are granted access to Bypass Unlock are registered by BlackBerry, and each issued a unique registration message. The message must be embedded in the application declaration, at build-time.

When to request reqistration

You should request access to the Bypass Unlock feature for your application towards the end of your initial implementation of the feature.

Procedure

Register your application for access to Bypass Unlock as follows.

1. Send a Bypass Unlock access request.

   Send an email message to the Bypass Unlock Registrar at BlackBerry, at the following address:
   BlackBerryDynamicsRegistrar@blackberry.com

   Include the following.

   - If there is an Android application, then the `settings.json` and `AndroidManifest.xml` files.
   - If there is an iOS application, then the Info.plist file.
   - The Application Policies definition XML file.
   - Completed Bypass Unlock request form, which will summarise the following:
     - Type of application.
     - What data appears on each Bypass screen. The screens will have been listed in the files mentioned above.
     - How the application user interface behaves differently if Bypass Unlock is allowed and disallowed by the enterprise.
     - Bypass Unlock options for the enterprise administrator and end user, if any.

   Templates for the email message and form text file are given below.

2. Answer any questions from the registrar.

   The registrar may send you questions by reply. In general, the answers to the questions should be added to the appropriate section in the request form text file.

   The registrar may also request changes in your application, for example removing or making optional the display of particular data items on a Bypass screen.

3. Build the application with modified application declarations from the registrar.

   When all changes have been made, the registrar will issue a *Bypass Unlock registration message*. The form of the message will be some signed data that you insert into your application's `settings.json` or Info.plist files. Some example registration messages are given below.

Templates

You can use the following as templates for your access request.

- Template for the access request email message.

To: BlackBerryDynamicsRegistrar@blackberry.com
Subject: Bypass Unlock access request

Hello Bypass Unlock Registrar,

Please can you grant access to the Bypass Unlock feature for my BlackBerry Dynamics application. The following files are attached to this message as part of the request.

- The `settings.json` and `AndroidManifest.xml` files from my application for Android.
- The Info.plist file from my application for iOS.
- The Application Policies definition XML file for my BlackBerry Dynamics application.
- A completed Bypass Unlock request form, in a plain text file.

Best regards, <Your Name>.
<Your contact details.>

- Template for the Bypass Unlock request form text file, mentioned in the email message template above.

Bypass Unlock Request Form

This text file contains the details for the request email message to which it is attached.

Type of application: <Examples: Voice over IP, or Business Dashboard.>

Data displayed in Bypass Unlock:

Screen: <Activity class name, for Android, or View Controller class name, for iOS. Example: InboundCallBypassViewController>
Data:
<List the data items displayed on the screen. If there are options for the enterprise administrator or end user, list the maximal set of data items here and note below what the options are. Example:

- Displays a notification that there is an inbound call.
- The number of the calling party, optionally.
- The name and company affiliation of the calling party, optionally.

Include the Screen and Data information for each Bypass Screen separately.>

User experience differences:
<State how the user experience differs if Bypass Unlock is allowed and disallowed by the enterprise. Example:
If Bypass Unlock is allowed, when the end user receives a phone call and the application is in the Idle locked state, a variant of the inbound call notification screen is displayed.>

Options:
<List the options available to the enterprise administrator and end user, if any. Example:
The enterprise administrator and the end user can choose which data items are displayed on the Bypass inbound call notification screen, see above. The end user can select not to display data items that are permitted by the enterprise administrator.>

Example Registration Messages

The following is a sample Bypass Unlock registration message for Android. It would be inserted into the application `settings.json` file. The signature has been shortened for convenience.

```
"GDPermissions": [
    {
        "Signature": "0b9aeff6...1ced54863754d9",
        "SignatureScheme": "RSAv1",
        "Permission":
        {
            "nativeApplicationId": "com.good.example.sdk.bypassunlock",
            "GDApplicationId": "com.good.example.sdk.bypassunlock",
            "BypassUnlockPermission": "1"
        }
    }
]
```

The following is a sample Bypass Unlock registration message for iOS. It would be inserted into the application Info.plist file. The signature has been shortened for convenience.

```
<key>GDPermissions</key>
<array>
    <dict>
        <key>Permission</key>
        <dict>
            <key>BypassUnlockPermission</key>
            <string>1</string>
            <key>GDApplicationId</key>
            <string>com.good.example.sdk.bypassunlock</string>
            <key>nativeApplicationId</key>
            <string>com.good.example.sdk.bypassunlock</string>
        </dict>
        <key>Signature</key>
        <string>0b9aeff...ced54863754d9</string>
        <key>SignatureScheme</key>
        <string>RSAv1</string>
    </dict>
</array>
```

When to make a subsequent registration request

The registration message generally remains valid through updates to your application and through new versions of the SDK. However, you will need to request access again if:

- The native application identifier changes, i.e. the package name, for Android, or the bundle identifier, for iOS.
- The entitlement identifier (generally known as the GD Application Identifier, or GD App ID) changes.
- A new Bypass screen is added.
- Additional data items are added to an existing Bypass screen.
- There is any other substantial change to the Bypass Unlock aspect of the application.

Discussion

Some points arising from the implementation tasks, above, are discussed in the following.

User Experience and User Interface

You could begin your implementation by deciding which parts of the user experience (UX) of your application will benefit from Bypass Unlock. For example, one part of a VoIP user experience that would benefit could be answering a call.

Each part of the UX to which Bypass Unlock will apply must be isolated in a separate screen in the user interface (UI). For the different mobile platforms, a screen corresponds to:

- An Activity, in an Android application.
- A View Controller, in an iOS application.

You might have to create new UI screens, if any part of the UX to which you will apply Bypass Unlock isn't already isolated on a separate screen, or set of screens.

The part of the UX to which Bypass Unlock will apply might already be on a separate screen, in which case you won't have to create any new screens. For example, in a VoIP application there might already be a separate Inbound Call screen.

Bypass Unlock Data

There are a number of approaches you can take to the presentation of enterprise data on a Bypass screen.

- No data on Bypass screens.

  The most secure approach could be to display no data on Bypass screens. This is simple but might lead to a worse UX in some applications. The following examples illustrate this approach.

  - The Inbound Call screen in a VoIP application alerts the user that there is an inbound call but doesn't include any information about the caller. The user would then have to make an uninformed decision to accept or reject the call.
  - An application for field engineers could have a Bypass screen, the Utilities screen, from which the user can open the device camera. If the user takes a picture, the image data is written to the secure store, and the application returns to the Utilities screen. There is no way for the user to view captured images from a Bypass screen, nor in the native photo gallery application.
  - An application for note-taking could have a Bypass screen, the Quick Note screen, on which the user can write themselves a text note. When the user saves the note, it is written to the secure store, and the screen is cleared. There is no way for the user to view saved notes from a Bypass screen.

- No enterprise data on Bypass screens.

  A less secure approach then displaying no data at all would be to display no data that came from the enterprise.

  The following examples illustrate this approach.

  - The Inbound Call screen in the previous example could display the caller's phone number. It wouldn't compare the caller's number to the numbers in the user's enterprise address book, so wouldn't display the name or company affiliation of the caller.
  - The Utilities screen in the previous example could display a count of the number of photos captured, and the time of the last capture.
  - The Quick Note screen in the previous example could display a count of the number of notes saved, and the time of the last save.

- Less valuable enterprise data on Bypass screens.

  A less secure approach than displaying no enterprise data would be to display enterprise data that is somehow less valuable than that in the main part of the application UI. Depending on what type of enterprise data the application handles, one or more of the following might apply.

  - Summarised data is less valuable than detailed data.
  - Old data is less valuable than new data.

  The following examples illustrate this approach.

  - An application for workflow could display the number of outstanding items in the user's inbox, and the number of items for which the user is awaiting actions by others.
  - An application with a project status dashboard displays information from yesterday, or from the last time the user authenticated.

- Limited enterprise data on Bypass screens.

  Another approach that is less secure than displaying no enterprise data would be to display a very limited subset of enterprise data.

  The following examples illustrate this approach.

  - The Inbound Call screen in the earlier examples could display the name of the caller, but not any other names.
  - The Utilities screen in the earlier examples could display the last photo taken, but not any others.
  - The workflow application in the previous example could display the newest item in the user's inbox, but not any other items.

Further options for administrators and end users

In case it isn't obvious which of these applies best to your application, you could give options to the enterprise administrator, or to the end user, or to both.

For example, the enterprise administrator could have an option to select what information is shown on the Inbound Call screen in the above examples. The end user could also have an option.

If you give options to both the administrator and the end user, the end user must be restricted to selecting a more secure setting than the administrator. For example, the administrator might select to show the phone number of an inbound caller in a VoIP application. The end user could still select to show nothing, but couldn't select to show the caller's name.

Options for the enterprise administrator would be implemented as Application Policy settings. Options for the end user would be implemented in the settings or preferences UI in the mobile application.

Bypass versions of screens

Whichever approach you take might require the creation of Bypass Unlock versions of existing UI screens.

For example, if the end user of a VoIP application happens to be authenticated and active when they receive a call, a full Inboud Call screen could be displayed. The screen could show the name and company affiliation of the caller, and could have obtained that information from enterprise data. But, if the application UI had been idle locked at the time of the call, a limited version of the screen could be shown instead. The full screen would be a Protected screen; the limited version would be a Bypass screen.

The application can track the following conditions:

- Idle lock is in effect.
- Bypass Unlock is allowed by the enterprise.

If both conditions are met at the time the call is received, the application would display the limited Bypass screen. Otherwise, it would display the full Protected screen. Note that there would be no point in displaying the limited screen if Bypass Unlock isn't allowed. The end user would have to authenticate to view the screen in that case, so they might as well be shown the full screen.

Bypass Unlock is optional

The enterprise will always have the option to disallow Bypass Unlock for your application. The UI of your application mustn't depend on Bypass Unlock always being allowed.

Transition from Bypass to Protected

If your application attempts to display a Protected screen when the UI is locked, the unlock screen will be superimposed. That still applies if a Bypass screen is current when the attempt to display a Protected screen is made. The transition from a Bypass screen to a Protected screen is a special case that should be handled by your application.

The transition from Bypass UI to Protected can be handled with the folowing implementation.

- Track the idle lock.

  Your application can be notified of state changes, including the expiry of the idle time out. The runtime dispatches an event every time the status changes. For the different mobile platforms:

  - Android applications can include a `GDAppEventListener` instance. When the idle time out expires, the listener `onGDEvent` method will be invoked and passed a `GDAppEvent` with the following characteristics:

    - The `getEventType()` accessor will return `GDAppEventNotAuthorized`.
    - The `getResultCode()` accessor will return `GDErrorIdleLockout`.

  - iOS applications can include a GDiOSDelegate instance. When the idle time out expires, the delegate `handleEvent:` function will be invoked and passed a `GDAppEvent` with the following property values:

    - `type GDAppEventNotAuthorized`.
    - `code GDErrorIdleLockout`.

  When your application is notified of idle lock, set an internal flag, and put one or both of the following into effect.

- Highlight options that will lead to transition.

  For example, show a padlock icon by any button on a Bypass screen that opens a Protected screen.

  If the Bypass screen is part of the main UI, and could be shown when there is no idle lock in effect, then the highlight should only be shown if there actually is an idle lock.

- Warn the user and give them a cancel option before transition.

  For example, if the user presses a button on a Bypass screen that would open a Protected screen, show a message like “Authentication will be required” with OK and Cancel options.

  The same advice applies to the warning as to the highlight in the previous point. A cancel option need only be offered if an idle lock actually is in effect.

In the scenario that the application UI is locked for inactivity, and a Bypass screen is open, and the user selects an option that would open a Protected screen, the unlock screen will be superimposed. There won't then be any option for the end user to cancel the unlock and return to the Bypass screen.

The above implementation is suggested as a way to avoid a poor user experience in these scenarios.

Authentication Delegation

Authentication of the end user can be delegated from one application to another. This feature, *authentication delegation*, is controlled by enterprise policy. If authentication delegation is in use, any delegating application might yield the foreground to the delegate when the idle time out expires. This could have an impact on an application that uses Bypass Unlock.

Consider the following scenario:

- An end user is running your application in foreground.
- Authentication delegation is specified by enterprise policy, to an application other than yours.
- The idle time out expires.

What will happen is that the other application will be brought to the foreground, and will show the unlock screen. If your application then attempts to show a Bypass screen, nothing will appear to the user, because the other application will be in foreground. Note that the scenario is unlikely to happen in typical usage.

If the application is in foreground, then the user will probably be interacting with it and the idle time out won't expire. Also, the idle time out would generally be longer than the device's native screen time out. This means that the application user interface will already be hidden when the idle time out expires.

You can reduce the likelihood of a poor user experience in the authentication delegation scenario, by tracking the native screen state. When your application is hidden by expiry of the native screen timer, immediately display a Bypass screen if there is one that is suitable to the current screen. When the hidden state of your application is cleared, check the idle lock.

- If the idle lock isn't in place, you can display the original screen that was open when your application was hidden.
- If the idle lock is in place, do nothing and leave the Bypass screen displayed.

Fingerprint Authentication

The product for iOS supports use of Touch ID for authentication of the end user, in certain situations and as controlled by enterprise policy settings. It is possible that the device will superimpose a prompt for the user to authenticate with a fingerprint even when a Bypass screen is displayed. This cannot be controlled by BlackBerry Dynamics, nor by your application code.

Enterprise Policy Setting

You must give the enterprise administrator the option to disallow Bypass Unlock in your application. Implement this as an Application Policy.

Implementation has the following parts.

- Create an Application Policies definition file, if your application doesn't have one.

  In case you require an introduction to the Application Policies feature, there is a technical brief on the application developer portal here: https://community.good.com/docs/DOC-1543

  The brief contains references to more technical and detailed resources, including the application programming interface references.

- Add a definition for the Bypass Unlock Policy setting to your Application Policy file.

  The policy setting must have a number of characteristics, as shown in the following XML snippet.

```
<!--
    Bypass Unlock Policy setting
    ============================
    The following setting controls whether Bypass Unlock is allowed or
    disallowed. It must be a checkbox setting.
-->
<setting name="GD_SDK_Security_AllowBypassUnlock" >
<!--
    The name attribute in the preceding tag must have the same value
    as the GDBypassUnlockPolicySetting in the settings.json or
    Info.plist file.
-->
    <checkbox>
        <key>GD_SDK_Security_AllowBypassUnlock</key>
        <!--
            The key must have the same value as the name attribute
            in the setting tag.
        -->
        <label
            >Allow parts of the user interface to be displayed when
            idle lock is in place.</label>
        <value>false</value>
        <!--
            The default, above, must be that Bypass Unlock isn't
            allowed.
        -->
    </checkbox>
</setting>
```

- Include the setting in the structural part of the Application Policies file.

  The structural part of the file, i.e. the top-level `pview` tag, defines how the settings will be arranged in the collection that is made available to the application code by the runtime. Every defined setting must appear in the structural part of the file.

  The structural part also defines how settings will appear to the enterprise administrator in the management console policy set editor. It is best practice to list the Bypass screens in description text, so that the enterprise administrator can make an informed decision about allowing Bypass Unlock.

  The following snippet is an example of the structural part of the definition.

```
<pview>

    <! -- Rest of the Application Policies go here. -->

    <!-- Separate tab for features, like Bypass Unlock. -->
    <pview type="tabbed" key="BlackBerryDynamicsFeatures">
        <title>BlackBerry Dynamics Features</title>
        <pe ref="GD_SDK_Security_AllowBypassUnlock" />
        <!--
            The ref attribute in the preceding tag must have the same
            value as the name attribute of the setting tag.

            Following is a list of Bypass screens, for the benefit of
            the enterprise administrator.
        -->
        <desc> - Incoming Call Screen</desc>
        <desc> - In-Call Screen </desc>
    </pview>
</pview>
```

- Check the policy before showing your Bypass screens.

  If your application attempts to display a Bypass screen but Bypass Unlock isn't allowed by the enterprise, then the unlock screen will be superimposed on it. Check the Bypass Unlock Policy setting with the normal Application Policy programming interface. For the different mobile platforms:

  - In an Android application, call the `GDAndroid getApplicationPolicy` or `getApplicationPolicyString` method to get the settings as a collection or as a JavaScript Object Notation (JSON) string, respectively.
  - In an iOS application, call the `GDiOS getApplicationPolicy` or `getApplicationPolicyString` functions to get the settings as a collection or JSON string, respectively.

  The policy setting will be in the setting that you defined in the Application Policy file. In the above example, it is the GD_SDK_Security_AllowBypassUnlock element.

Designation of screens

Designate the Bypass screens in your application as follows.

- In an Android application, the designation is made in the manifest. In the activity tag of each Bypass screen, insert a `meta-data` tag with the following characteristics:
  - `android:name="com.good.gd.bypassunlock"`
  - `android:value="true"`
- In an iOS application, the designation is made in the Info.plist file, in a property with the following characteristics:
  - Key: `GDBypassUnlockViewControllers`.
  - Type: Array of values, one per Bypass screen.
  - Array element type: String containing the class name of a View Controller that represents a Bypass screen.

There are code snippets in the Implementation Essentials, above.

Sample Applications

Each of the SDK distributions that supports Bypass Unlock comes with a sample application. The applications are pre-registered with access to the feature. They can be used as starting points for implementation in your application.

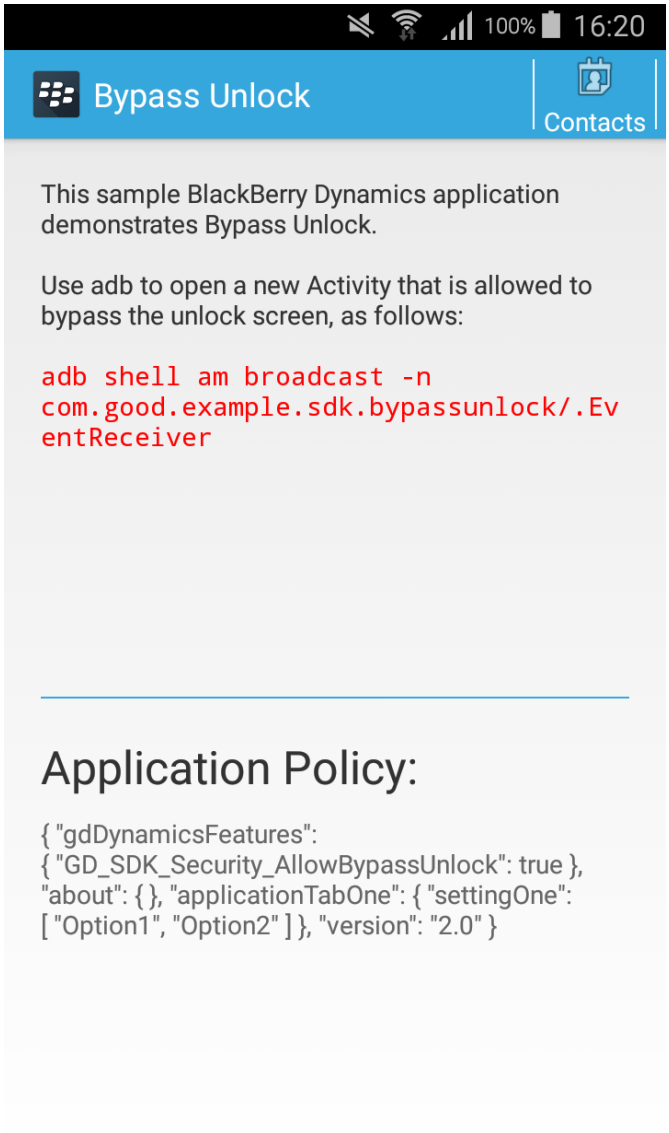
Each of the sample code projects includes an Application Policies definition XML file.

Sample Bypass Unlock Application for Android

The following screen capture images are from the Bypass Unlock sample application for Android.

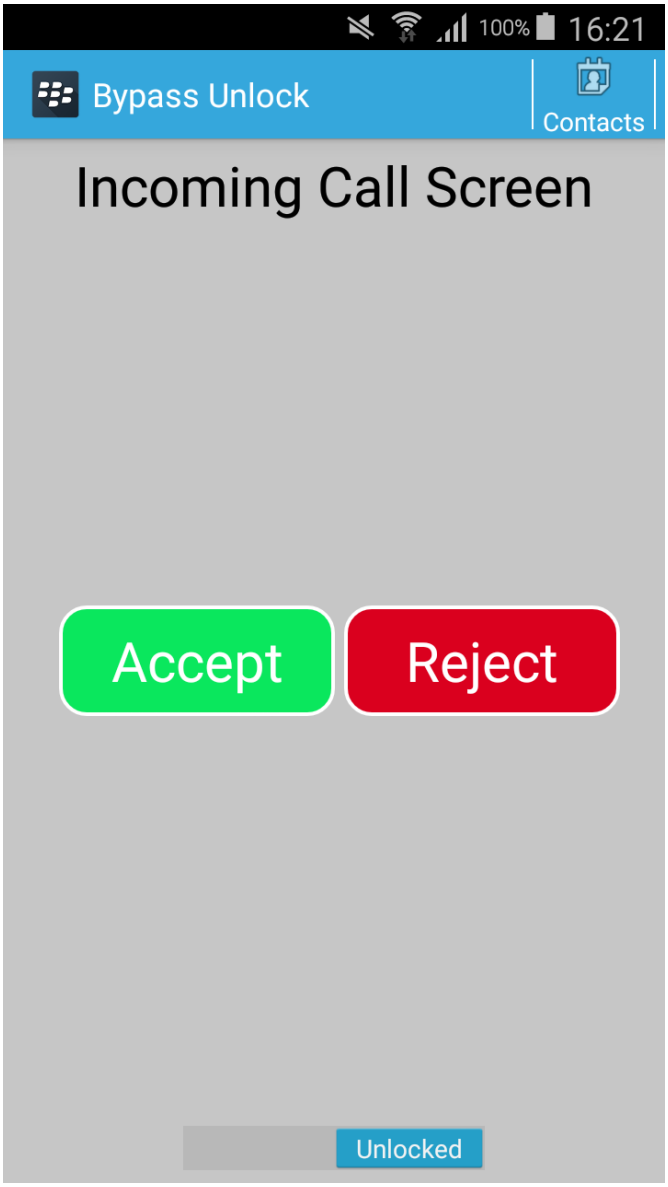
- MainActivity screen.

  The MainActivity is a Protected screen. It gives instructions how to demonstrate the Bypass Unlock feature in the application, and shows a diagnostic dump of the current application policy setting.



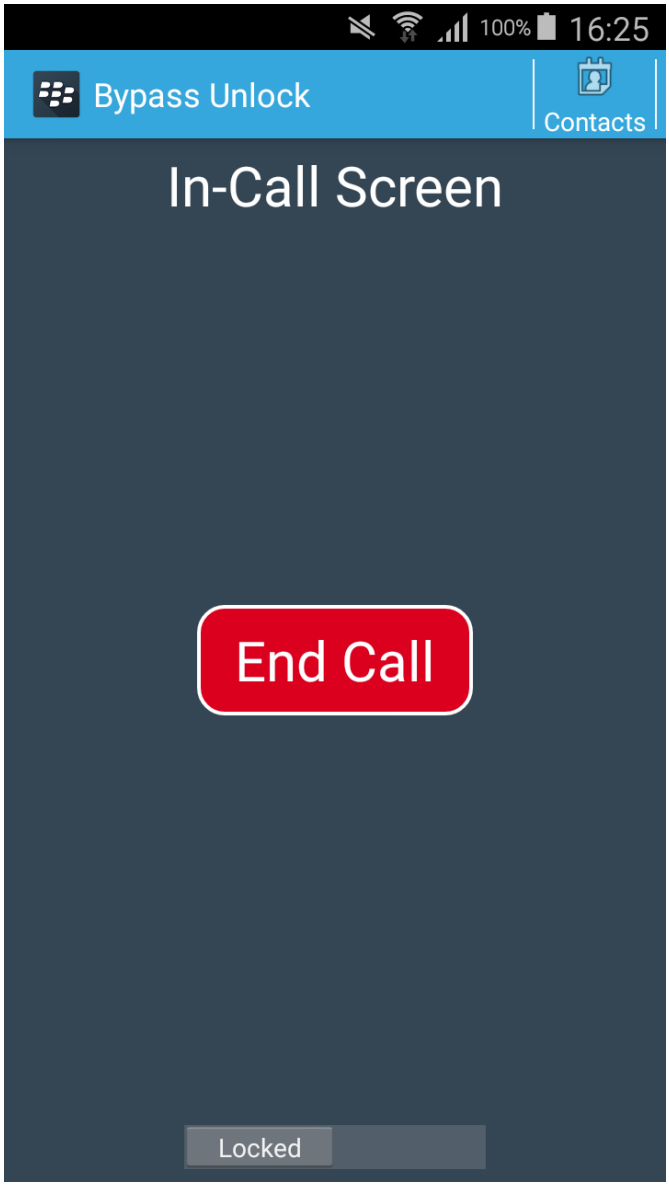
- IncomingEventActivity screen.

  The IncomingEventActivity is a Bypass screen. It demonstrates notification of the end user during idle lock. The switch at the bottom of the screen shows the idle lock state.



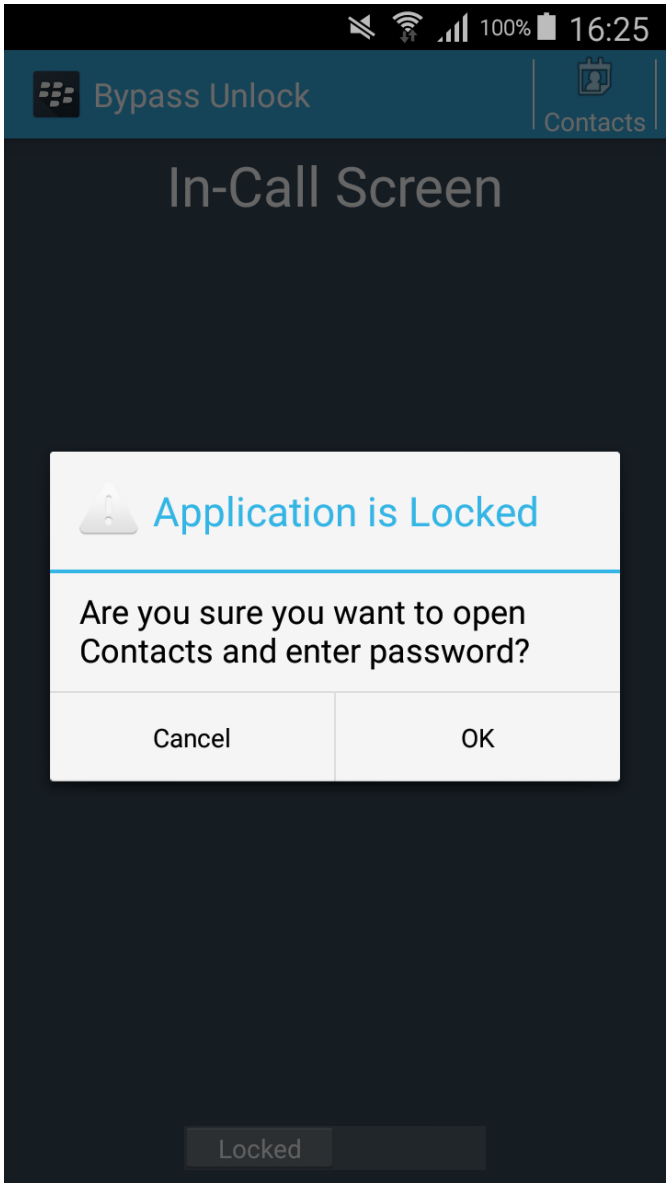
- InCallActivity screen.

  The InCallActivity is also a Bypass screen. It demonstrates offering the end user an action when the application is locked. The switch at the bottom of the screen shows the idle lock state. Note that there is an option for the user to open their Contacts.



- InCallActivity warning before attempting to open Contacts.

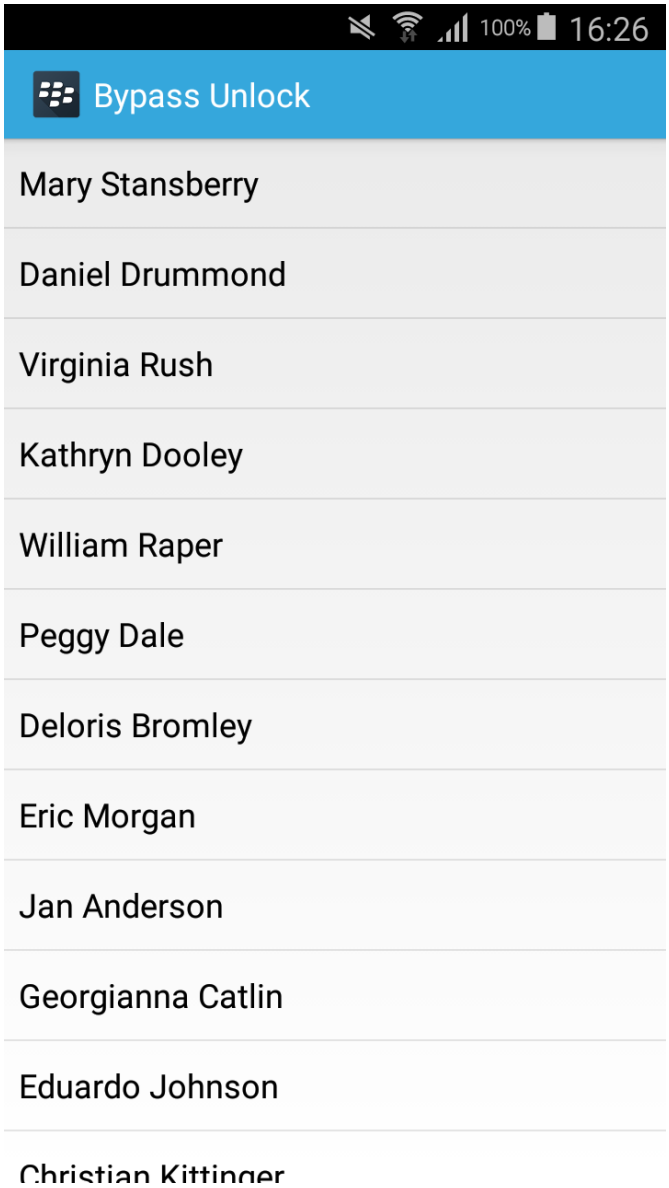
  If the user selects to open their Contacts from the InCallActivity screen, and the application is idle locked, then a warning is displayed. This is shown in the following screen capture image.



  If the user cancels, the application returns to the InCallActivity screen. If the user proceeds, they will have to authenticate. The user's Contacts won't be displayed unless the user is authenticated.

- ContactsActivity screen.

  The ContactsActivity is a Protected screen. It displays the user's Contacts.

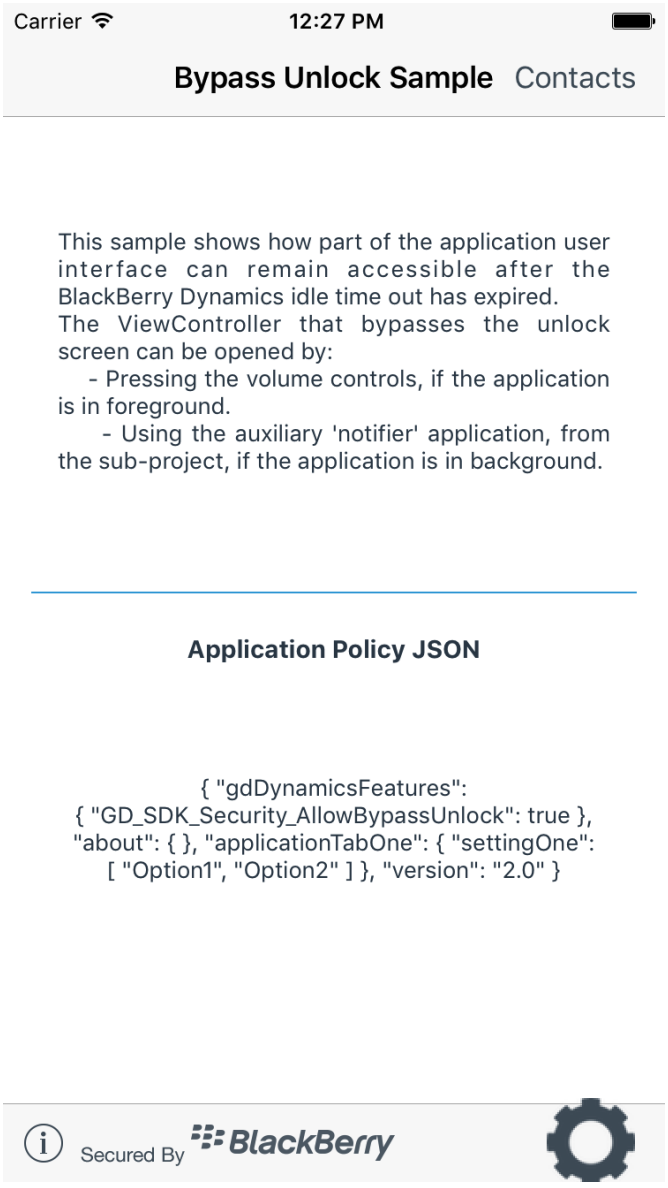


Sample Bypass Unlock Application for iOS

The following screen capture images are from the Bypass Unlock sample application for iOS.

- Main screen.

  The Main screen is Protected. It gives instructions how to demonstrate the Bypass Unlock feature in the application, and shows a diagnostic dump of the current application policy setting.



- Incoming Call screen.

  The Incoming Call screen is a Bypass screen. It demonstrates notification of the end user during idle lock.

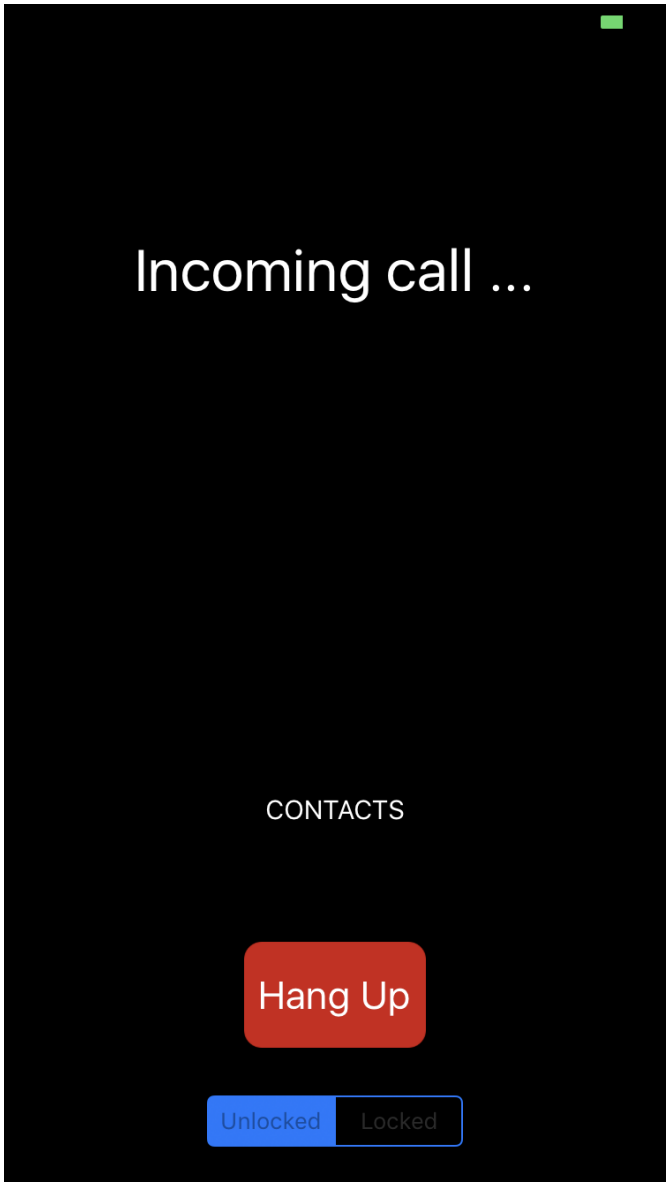
  This screen is displayed in the sample application by the following code:
  `[[BPCallManager sharedManager] simulateCall]`

  The switch at the bottom of the screen shows the idle lock state. This is tracked by receipt of the `GDErrorIdleLockout` event in the `AppDelegate` instance, and then reflected in a property of the `sharedManager` object. The first authentication of the end user since start-up is tracked in another property of the same object. The object and properties can be accessed as follows:

  - Idle locked state: `[BPCallManager sharedManager].appIdleLocked`
  - Authenticated state: `[BPCallManager sharedManager].appStarted`

- In Call screen.

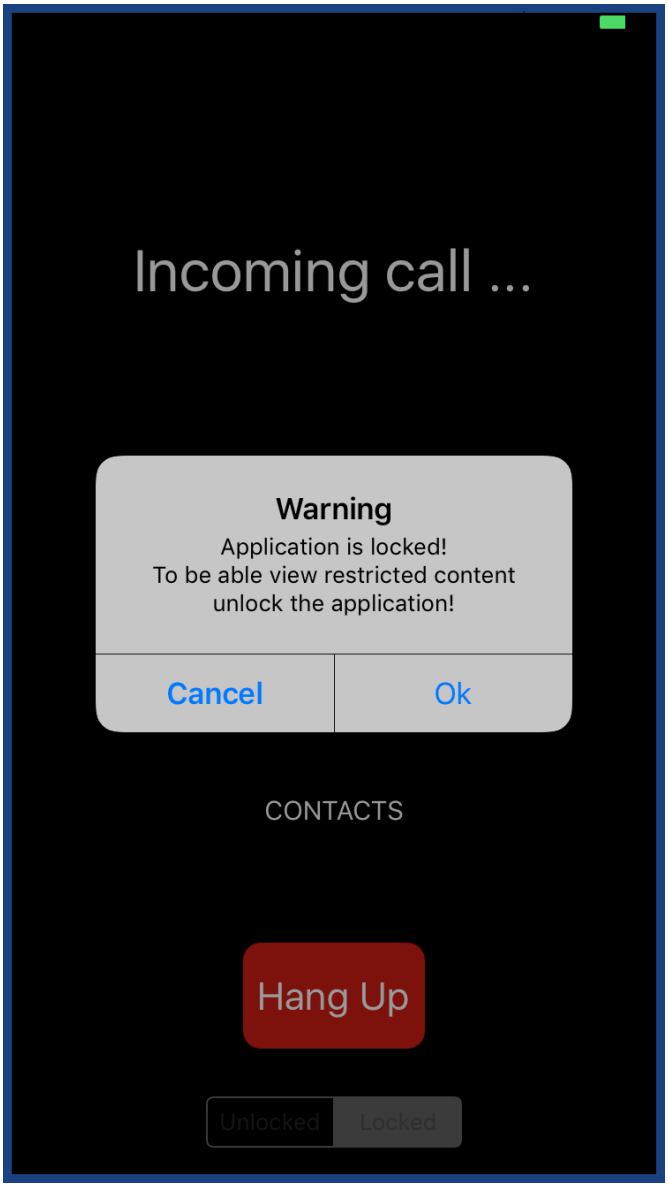
  The In Call screen is also a Bypass screen. It demonstrates offering the end user an action when the application is locked. The switch at the bottom of the screen shows the idle lock state. Note that there is an option for the user to open their Contacts.



  In this screen capture, the application happens to be unlocked.

- In Call warning before attempting to open Contacts.

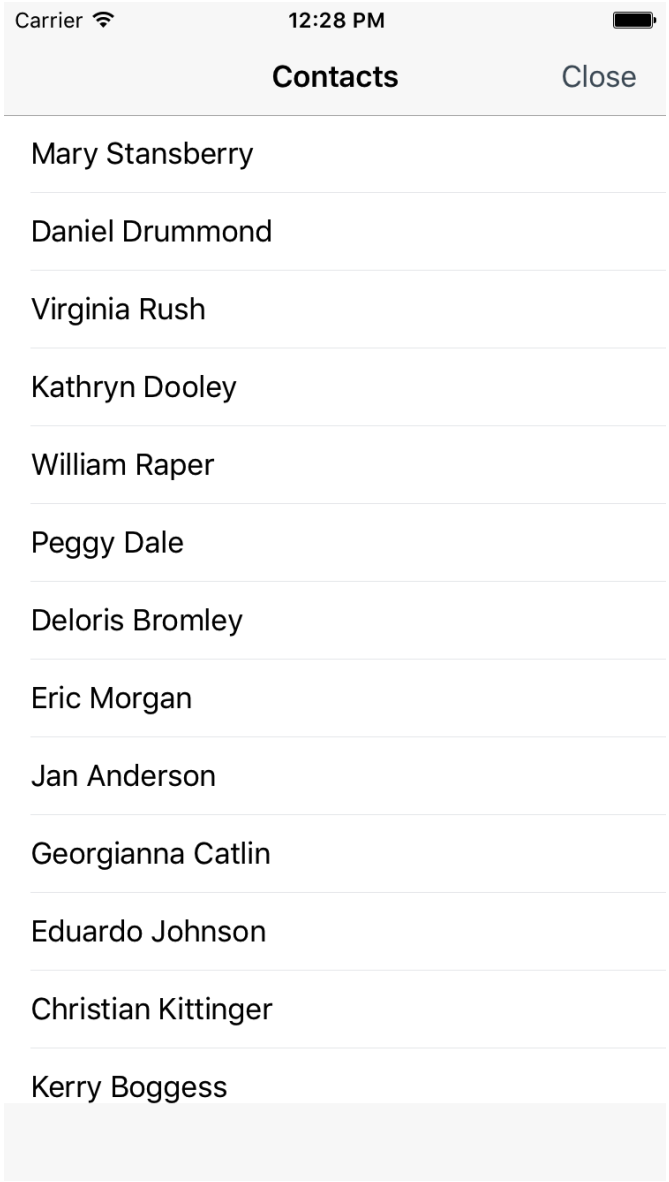
  If the user selects to open their Contacts from the In Call screen, and the application is idle locked, then a warning is displayed. This is shown in the following screen capture image.



  If the user cancels, the application returns to the In Call screen. If the user proceeds, they will have to authenticate. The user's Contacts won't be displayed unless the user is authenticated.

- Contacts screen.

  The Contacts screen is Protected. It displays the user's Contacts.



  This screen is represented by the `BPContactsViewController` class. There are two ways to open this screen:

  - From the navigation bar, by pressing the Contacts button, if the application isn't locked.
  - From the Contacts button on the In Call screen. If the application is locked then the user will have to authenticate.

Legal Notice

Legal Information