The Open Mobile Provisioning API

VERSION 3.2 JUNE 2014

Corporate Headquarters
iPass Inc.
3800 Bridge Parkway
Redwood Shores, CA 94065 USA

www.ipass.com
+1 650-232-4100
+1 650-232-0227 fx

TABLE OF CONTENTS

Understanding Open Mobile Provisioning

The *provisioning* process enables Open Mobile apps to connect to iPass networks. Provisioning is performed after installation, and includes it two separate but related operations: *activation* and *settings configuration.* In addition, Open Mobile supports the *re-provisioning* of devices that have been previously activated or configured.

Activation

The Open Mobile app comprises the Open Mobile software and a *profile* that dictates its service capabilities. The profile consists of parameters and settings that determine the behavior of the software. These parameters and settings can include, but are not limited to, lists of accessible networks, policy settings, and valid options for configuring authentication parameters.

Different profiles can be used to address the needs of different segments of a user base. For example, one profile could enable Wi-Fi connections, specify permitted networks for connection, and require that a user enter a username and password in order to connect. A second profile could specify entirely different networks, and require that users must enter a username, password and domain to connect.

Profiles are created on the Open Mobile portal by an Open Mobile administrator.

When Open Mobile is downloaded and installed from an app store or other source, it does not include a profile. When the *activation* process is initiated, the user may enter a Activation Code (Profile ID and a PIN, if a PIN has been assigned to the profile). Open Mobile makes a connection to the Internet, downloads the identified profile from the Open Mobile provisioning server, and then writes the profile parameters to the device. The activation process can be initiated through user inputs (either through the Activation Wizard or the Activation Code) or invocation of the provisioning API. Activation is a one-time event for unactivated apps.

An app that has not been activated (and has no profile) is not usable for connections, but other minor features of the app may be available.

Service Settings Configuration

After activation, the user can configure a group of service settings for identification, authentication, and control of the behavior of the app. These service settings include:

- **Credentials:** Comprise of the user’s username and password, which are used for identification and authentication of the specific user.
- **Service Attributes:** Comprise of the domain and prefix, which identify the organization to which the user belongs so that iPass can properly forward authentication attempts to the proper authentication and authorization (AAA) server. Domain and prefix may be actively configured by the user, configured by the provisioning API, or determined by default from the Open Mobile profile.
- **Preferences:** Enable features that make the application easier to use. There is currently a single service preference setting: *Auto-login*. If set and the user’s credentials are saved, Auto-login will enable automatic logins to any iPass network in range.

A client without these settings configured and saved will need them configured by the user before each connection. For example, if the user's credentials are not saved, the user is prompted to enter them manually before each connection attempt.

Service setting configuration may be performed on a device any number of times.

Re-provisioning

In addition to initial activation and settings configuration, Open Mobile supports re-provisioning, which includes reactivation or reconfiguration of an active device. Re-provisioning may be performed on a device any number of times.

- **Reactivation:** The client may be re-activated using a new profile ID and PIN, which results in complete replacement of the profile and its associated configuration.
- **Reconfiguration:** Existing service settings (such as domain) may be completely or partially reconfigured. In addition, a setting type that was not used by the client prior to re-provisioning could be added to a client installation. For example, if routing prefix was not used in the client's initial installation, but was later required, an Open Mobile client could be re-provisioned to accommodate both the new setting and a value for the setting.

Depending on the content of a new profile, re-provisioning may erase some existing client settings. Re-provisioning can also fail (report errors) if the new profile is incompatible with existing client settings.

Using the Provisioning API

The Open Mobile Provisioning API provides a programmatic means for third-party applications to perform provisioning of the Open Mobile application (Open Mobile v2.4 for Android, v2.2 for iOS, and later versions of both).

Logically, the Provisioning API provides a means to perform automated profile activation or service settings configuration. These functions are normally performed after inputs from the user. Both of these operations can be completed with a single call to the provisioning API, reducing or removing completely any requirements for user interaction. Alternatively, you can perform any subset of the provisioning function, including re-provisioning a device partially or completely, according to your security management policies.

The API should only be invoked from an application running on the same mobile device targeted for provisioning.

> *The developer of a new provisioning application should be familiar with the Open Mobile activation and settings UIs, as well as profile and option definitions on the Open Mobile service portal.*

Supported Use Cases

The application can be designed to support any or all of these operations:

- Automation of the complete provisioning process for a newly downloaded client, so that no user inputs are required. In this scenario, the user would download and install the Open Mobile to a device, and the provisioning application would automatically activate Open Mobile and then configure appropriate user settings based on your organization’s preferences without user intervention.
- Activation or configuration of a device.
- Reactivation of a device that already has an Open Mobile profile.
- Configuration (or re-configuration) of one, some, or all service settings.

This list is not inclusive and can extend to other use cases as well.

A well-designed provisioning application can shield the user from the complexity and possible frustration of manual entry of activation profile, PIN, and settings to provide a complete and versatile provisioning experience.

Provisioning Application Goals

In general, a new provisioning application will:

- Enable provisioning either as a single operation or as a series of operations, as long as intermediate states of provisioning do not result in an invalid client configuration.
- Ignore the presence or absence of provisioning parameters when the profile contents imply a single, unambiguous configuration.
- Prevent configuration of Open Mobile in a way that is internally inconsistent (and return a status code indicating an attempted inconsistency).
- Prevent inadvertent *deconfiguration* of Open Mobile. Deconfiguration is an operation that changes the program configuration in such a way that the app is no longer able to establish connections without user intervention to provide Setting values.
- Manage the delivery of the activation and authentication parameters to the target device. Open Mobile will

not convey the parameters to other applications, record the password parameter in any software instrumentation log, or convey it to any other system, except as a part of a network authentication attempt. The status of a network authentication attempt will be reported to the third party application by a means appropriate to the platform.

Soft Configured Settings

To simplify the provisioning process, the Open Mobile provisioning facility implicitly performs the following automatic configuration operations, when appropriate and will not require execution as part of your provisioning app.

- Configures (or reconfigures) the domain setting to the first domain option identified in the profile when a provisioned profile includes one or more domain options and the current settings do not include a domain value. This occurs during the initial activation if the domain parameter is omitted and whenever the domain value is configured as a null (blank) value.
- Removes the configured password setting when a new profile is provisioned that prohibits the saving of password.
- Removes the prefix setting when a new profile is configured that does not include a prefix option.
- On Android:
  - Sets auto-login to No on Android during the initial activation.

    Auto-login is labeled Auto-connect in the Android user interface.
- On iOS:
  - Sets auto-login to No when a profile re-provision activates a profile that forbids user control of the auto-login feature.
  - Sets auto-login to Yes during the initial activation when a profile is provisioned that allows user control of the auto-login feature.

These automatic operations are exactly parallel to those performed during manual configuration.

Error Codes

Because the provisioning facility may be used to either initially configure, or partially re-configure a client, error indications relate to the *effective configuration* of the client. A client's effective configuration is defined as:

1. The combination of the current client configuration (profile and settings).
2. The effective changes requested by the provisioning operation.
3. Settings which are configured implicitly by the soft configuration rules, described earlier.

The provisioning facility determines the new effective configuration, and then evaluates its contents and settings to determine if an incompatible configuration has been requested. Error codes report the first inconsistent condition detected during the configuration validation.

The provisioning facility is designed to be accommodating to provisioning attempts that make unusual use of the configuration parameters. In general, a provisioning request that does not attempt to create an incompatibility between the client settings (such as password, username, or domain) and the provisioned profile will succeed, even if the user explicitly sets values that might be implicit (soft-configured) in the profile.

In some cases, excluding a setting value may have the same effect as specifying a null value for the setting. Similarly, setting a parameter to a value that matches an implicit value (as described above) will succeed. For example, this could include setting a value for domain that exactly matches the single, non-editable domain value allowed by the profile. As a result, there may be any number of valid ways to request a particular configuration.

Note that each call to the interface will either execute completely, or fail completely. In the event of a failure, there will be no net change to the application state. No provisioning operation will be left half-completed.

iOS Provisioning Facility

iOS supports a single method of inter-process communication, which is based on URL protocol handlers. The provisioning API defines the scheme used to by your calls to the provisioning function.

To receive the operational status from a provisioning request, your app must create its own protocol scheme. This scheme must be conveyed to Open Mobile in the provisioning call, which Open Mobile will then use to convey the operation status.

App Focus

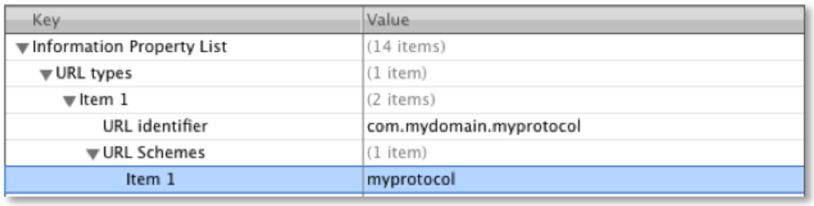
iOS does not facilitate reliable bi-directional inter-process communication. To request the iOS provisioning facility, the requesting app must be active in the foreground. Attempts to invoke the provisioning facility from background tasks fail quietly (that is, an openURL operation reports no error, but iOS quietly ignores the requested operation.)

Likewise, for Open Mobile to return the operation status to the requesting program, Open Mobile *must* be active in the foreground during the completion of the provisioning operation. If the user backgrounds Open Mobile while the provisioning operation is in progress, the provisioning operation will complete normally, but the resulting status will be quietly discarded by iOS when Open Mobile attempts to return it. As a result, users should be advised not to background Open Mobile until any provisioning tasks are completed by the third-party app.

Creating a URL Protocol Handler

Creating a URL protocol handler in your program is relatively straightforward for iOS applications, requiring the following two items:

- A declaration of a serviced URL scheme in the `info.plist`, as shown in Figure 1:



Key	Value
▼Information Property List	(14 items)
▼URL types	(1 item)
▼Item 1	(2 items)
URL identifier	com.mydomain.myprotocol
▼URL Schemes	(1 item)
Item 1	myprotocol

Figure 1

- A suitable protocol handler function in the app delegate, to pass the URL on to a class that does the work:

```
- (BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url {
     [ServiceManager handleURL:url];
     return YES;
}
```

On iOS devices, the provisioning function may be requested using properly formatted URLs invoked through Safari on the target device (for example, by clicking on web page links on your web site). Such invocations cannot directly report the status of the operation, except through the status displayed by Open Mobile at the completion of the request. Since sensitive parameters (such as password) may be specified in your provisioning request URL, use care in the design of any provisioning web page links to limit the visibility of such parameters.

iOS Provisioning URL Format

A valid URL for an iOS provisioning facility has the following form:

```
ipass://provision?pid=<profile id>[&pin=<pin>][&test=<N/Y>][username=<profile
id>][&password=<pwd>][&domain=<auth domain>][prefix=<user
prefix>][autologin={N,Y}][&resulturlscheme=<status URL scheme>][&obfuscation={N,Y}]
```

The parameters of this URL are described in Table 1. Note that Open Mobile implements a protocol scheme handler for the `ipass` scheme to enable your requests to reach Open Mobile.

Table 1

Parameter	Description
`pid`	Numeric profile ID.
`pin`	PIN (if there is one). If no PIN is associated with the profile, this parameter may be omitted, or the PIN value may be a null length string.
`test`	Indicates whether the profile is a test mode profile. Valid values are N (no) or Y (yes). • If the parameter is omitted, the setting is assumed to be N, indicating that it references a production mode profile. • Typically, this value will be omitted, except for special versions of a third party application that may be used by an IT administrator to test activation with a new revision of a profile, before it is published for devices in the field.
`username`	Username for the device user.
`password`	Password for the device user.
`domain`	Authentication domain for the user (if any). This value may be implied by the profile and not required for the provisioning operation.
`prefix`	Routing prefix to be prefixed to the NAI (if any is defined in the customer account).
`autologin profile`	Indicates whether the auto-login option is to be enabled on the user device. Valid values are N (no) or Y (yes). The provisioned profile specifies whether this value may be configured. • If the profile prohibits user control of this feature, the value is always implicitly set to N (the client will not automatically login). • If user control is allowed by the profile, the default value is Y. If this parameter is absent, either no change to the current configured setting will occur, or Open Mobile will implicitly set the value based on the default value determined by the profile and Open Mobile client type (as described above in Soft Configured Settings).
`resulturlscheme`	URL scheme implemented by your app to allow it to receive the operational status. For example, if the `resulturlscheme` identified by your application is `myprotocol`, Open Mobile will invoke the third party application's handler by opening a URL of the following form: `myprotocol://provision?result=<operation status identifier>, where` • `myprotocol` is the scheme or protocol specifier for the status handler in the third-party application. • `provision` indicates the invoked operation for which this is the completion status. • `result is` the string identifying the outcome of the activation attempt. This string may be any of the results from Table 2. If `resulturlscheme` is omitted or is null, no status will be received by your application upon completion of the provisioning attempt.
`obfuscation`	In the URL Method, it may be desirable for security reasons to obfuscate the password field. This parameter, when set to Y, will obfuscate the password with AES-256 using a pre-defined shared key. A Java program will be provided to

Parameter	Description
	obfuscate the password. Please contact your account team for access to this password obfuscation tool. *If the parameter is omitted, the setting is assumed to be N.* *This option is applicable to Android and iOS only.*

Table 2

Result Code	Description
0	The operation succeeded.
1	The operation failed. The operation could not be attempted because an activation or provisioning operation is already in progress (perhaps by another application).
2	An unexpected internal error occurred. Open Mobile has internally recorded exception information in its status log. Instruct the user to perform the Send Logs operation in Open Mobile, and then forward the received Open Mobile log to iPass through a customer service ticket.
10	The operation failed. A blank Profile ID was specified. (A valid Profile ID is a decimal value.)
11	The operation failed. The auto-login or test setting was specified as a value other than 'Y' or 'N'.
20	The operation failed. The effective domain configuration is incompatible with the provisioned profile.
21	The operation failed. The effective user prefix configuration is incompatible with the provisioned profile.
22	The operation failed. The requested operation asserts the autologin feature but: • The effective configuration of the client is missing a valid username, password, or domain, or, alternatively • The provisioned profile prohibits user control of the feature and the parameter specifies a setting of Y
23	The operation failed for one of the following reasons: • The requested username or password setting exceeds the maximum length of those values (character length limits for username: 100 characters, for password: 64 characters). • A valid password value was supplied, but the effective configuration is missing a username or domain setting. • A valid password value was supplied, but the provisioned profile prohibits the saving of the password in Open Mobile (and requires the user to input it each time a connection is made).
30	The operation failed. Communication with the server was interrupted.
31	The operation failed. The activation service has reported a transient error. Try the operation again later.
32	The operation failed. The device does not have a connection to the Internet. A connection is required. Establish an Internet connection and retry the operation.
33	The operation failed. The Profile ID and PIN (if any) does not select an available profile for the client type (iOS) in the indicated mode (production or test). This error may occur for one of the following reasons: • The Profile ID and PIN specify a profile that is not an iOS client profile, and no iOS Favorite profile has been created for the company by the company's Service Administrator. • The PIN is omitted for a profile that requires a PIN.

Omitted Parameters

If a provisioning parameter is omitted, no previously provisioned value for the parameter will be replaced and the value will persist in the client configuration. If the application needs to erase any previously provisioned value for a parameter, a zero-length value string should be specified.

It is not required that a username, password, domain or prefix be provisioned. However, if the resulting configuration is missing a username, password, domain, or prefix, the provisioning operation will succeed but the client will not be immediately serviceable without a user manually supplying any missing parameters through the Open Mobile Settings dialog. Auto-login cannot function without all necessary settings, whether supplied by provisioning or by the user.

.

Android Provisioning Facility

There are two ways to send provisioning request to Open Mobile for Android.

URL Method

The first method works likes a URL, which can be used as a link. The link is should be created in the format specified in the section Android Provisioning URL Format. It is typically embedded in a web page or HTML file that can be distributed through a Mobile Device Management (MDM) software or hosted on an internal web server. The user needs to install Open Mobile and tap the link to provision the Open Mobile client.

An extra ".com" is required after clientx://provision in the URL Method (and not in the Intent Method).

Intent Method

In the second method, Open Mobile for Android will expose an implicit *intent* for provisioning a new service component as shown below. It is started when an application component of the third party app starts by calling `startservice(intent)`. The service will run in the background until the provisioning either has completed successfully or has failed due to an error.

Intents are messages that enable activities, services, and broadcast receivers. Intents can be used for both intra- and inter-application communication to invoke application components. The primary parameters in an intent are the action (the general process to be performed) and data (the information to operate on).

Shown here is a method invoking the Open Mobile provisioning service for activation by a third party application:

```
Intent provisionIntent = new Intent("android.intent.action.VIEW");
Uri uri = Uri.parse("clientx://provision?pid=112665&pin=1111&test=y");
provisionIntent.setData(uri);
getApplicationContext().startService(provisionIntent);
```

Receiving the Provisioning Request Result Code

A third-party application must create an Android BroadcastReceiver to receive the result of the provisioning operation. The receiver should register for the action `client_x.android.prov.result.` If the user does not register the BroadcastReceiver before invoking the provisioning operation, the requesting program cannot synchronize with the completion of the operation, nor can the result code be obtained.

The third-party app can statically publish an implementation through the `<receiver>` tag in `AndroidManifest.xml.` Shown here is an example of publishing Broadcast Receiver in `AndroidManifest.xml` to receive the result code:

```
<receiver android:name=".ProvisionResultReceiver">
    <intent-filter>
          <action android:name="client_x.android.prov.result"></action>
    </intent-filter>
</receiver>
```

If you publish the receiver in AndroidManifest.xml, then your application will not need to register or unregister receivers.

After declaring your BroadcastReceiver, create your receiver's service class as shown below, which will recover the provisioning status code:

```
// Declare your Broadcast receiver class
private class ProvisionResultReceiver extends BroadcastReceiver{
   @Override
   public void onReceive(Context context, Intent intent) {
   // Retrieve the status code
   int status = intent.getIntExtra("EXTRA_RESULT", -1);
   // do appropriate action
   }
};
```

Provisioning Result

The Open Mobile app broadcasts the result of the provisioning attempt to the third party app using this method:

```
Intent provisionStatusIntent = new Intent("client_x.android.prov.result");
provisionStatusIntent.putExtra("EXTRA_RESULT", operationStatus);
sendBroadcast(activationStatusIntent);
```

Android Provisioning URL Format

A valid URL for an Android provisioning facility has the following form:

```
clientx://provision.com?pid=<profile id>[&pin=<pin>][&test=<N/Y>][username=<profile
id>][&password=<pwd>][&domain=<auth domain>][&prefix=<user
prefix>][&autologin={N,Y}][&silentmode={N,Y}][&obfuscation={N,Y}]
```

The parameters of this URL are described in Table 3 below and are case-sensitive. All Boolean parameters that are omitted have a default value of N (No).

Note that ".com" is required after the keyword "provision" for the URL Method.

Table 3

Parameter	Description
`pid`	**Numeric profile ID.**
`pin`	**PIN (if there is one). If no PIN is associated with the profile, this parameter may be omitted, or the PIN value may be a null length string.**
`test`	**Indicates whether the profile is a test mode profile. Valid values are N (no) or Y (yes).** **Typically, this value will be omitted, except for special versions of a third party application that may be used by an IT administrator to test activation with a new revision of a profile, before it is published for devices in the field.** *If the parameter is omitted, the setting is assumed to be N, indicating that it references a production mode profile.*
`username`	**Username for the device user.**
`password`	**Password for the device user.**
`domain`	**Authentication domain for the user (if any). This value may be implied by the profile and not required for the provisioning operation.**
`prefix`	**Routing prefix to be prefixed to the NAI (if any is defined in the customer account).**
`autologin`	**Determines whether the auto-connect option is set on the user device. Valid values are 'N' (no) or 'Y' (yes). If this option is absent, no change to the current configured setting (if any) will occur. Open Mobile defaults this setting to N if the setting is not provisioned and is not specified by the Open Mobile profile.** *If the active profile forbids the saving of the user password, this setting has no effect on the behavior of Open Mobile. In addition, the auto-connect option will not appear in the Open Mobile user interface (and its effective setting is "N") for Android releases before v2.6. After v2.6, the effective setting of auto-connect defaults to a setting specified by the Open Mobile profile. If the profile forbids user control of auto-connect behavior, this setting is ignored and the effective auto-connect value is determined by the Open Mobile profile.*
`silentmode`	**When set to Y, Open Mobile will not be brought to foreground during provisioning. Instead a notification will be posted to indicate the outcome of provisioning. This option is suited for a deployment scenarios where users do not need to see the individual app being provisioned.** *If the parameter is omitted, the setting is assumed to be N.* *This option is applicable to Android only.*
`obfuscation`	**In the URL Method, it may be desirable for security reasons to obfuscate the password field. This parameter, when set to Y, will obfuscate the password with AES-256 using a pre-defined shared key. A Java program will be provided to obfuscate the password. See Utility to obfuscate password for an example.** *If the parameter is omitted, the setting is assumed to be N.* *This option is applicable to Android and iOS only.*

The result of the provisioning operation is broadcasted to the `BroadcastReceiver` of your application, as described in *Receiving the Provisioning Request Result Code*. The third party application must query for EXTRA_RESULT of integer type from the intent to receive the status code, as described in Table 4.

The URL method will not report the status code but there will be an on-screen dialog indicating success or failure.

Table 4

Result Code	Description
0	The operation succeeded.
1	The operation failed. The operation could not be attempted because an activation or provisioning operation is already in progress (perhaps by another program).
2	An unexpected internal error occurred. Open Mobile has internally recorded exception information in its status log. Instruct the user to perform a Send Logs operation in Open Mobile, and then forward the received Open Mobile log to iPass through a customer service ticket.
10	The operation failed. A blank Profile ID was specified. (A valid Profile ID is a decimal value.)
11	The operation failed. An invalid setting for auto-login or test was specified (not ‘Y’ or ‘N’).
20	The operation failed. The effective domain configuration is incompatible with the provisioned profile.
21	The operation failed. The effective user prefix configuration is incompatible with the provisioned profile.
23	The operation failed for one of the following reasons: • The requested username or password setting exceeds the maximum length of those values (the username limit is 100 characters and the password limit is 64 characters). • A valid password value was supplied, but the effective configuration is missing a username or domain setting. • A valid password value was supplied, but the provisioned profile prohibits the saving of the password in Open Mobile (and requires the user to input it each time a connection is made).
30	The operation failed. Communication with the server was interrupted.
31	The operation failed. The activation service has reported a transient error. Try the operation again later.
32	The operation failed. The device does not have a connection to the internet. A connection is required. Establish an Internet connection and retry the operation.
33	The operation failed. The Profile ID and PIN (if any) does not select an available profile for the client type (Android) in the indicated mode (production or test). This error may occur for one of the following reasons: • The Profile ID and PIN specify a profile that is not an iOS client profile, and no iOS Favorite profile has been created for the company by the company’s Service Administrator. • The PIN is omitted for a profile that requires a PIN.

Omitted Parameters

If a provisioning parameter is omitted, any previously provisioned value for the parameter will not be replaced, and will persist in the client configuration. If the application needs to erase any previously provisioned value for a parameter, a zero-length value string should be specified.

Open Mobile will not permanently save a password configuration until domain, username, and password are assigned a value.

When a password value is specified, the operation will fail unless *all three values* are configured (either explicitly in the current provisioning request, or are present from an earlier provisioning operation). It is not required that a password be provisioned. However, if the configuration contains only a domain or username, the provisioning operation will succeed but the client will not be immediately serviceable without user inputs, using the Open Mobile **Settings** dialog to supply the missing parameters.

Provisioning Examples

This section contains examples of how to create a third-party application to accomplish one of the possible provisioning tasks for Open Mobile. These examples are not intended to describe the only possible uses of the API, but are merely good-practice illustrations.

> *For Android, the provision keyword after clientx may require ".com" depending on the delivery mechanism. The Intent Method DOES NOT require ".com" and the URL Method DOES require ".com". In the following examples, the URL method is assumed and ".com" is included.*

Complete Provisioning (Activation + Configuration)

This example illustrates a complete, "one-shot" provisioning solution, which automates the process of both initial activation and service settings. Following a successful provisioning, the device is ready to connect to iPass networks without further user configuration.

iOS

```
ipass://provision?pid=9945&username=jsmalley&password=doggylove2&domain=mycorp.com
```

Android

```
clientx://provision.com?pid=9945&username=jsmalley&password=doggylove2&domain=mycorp.co
m
```

Simple Activation

To activate a device, invoke the provisioning API, specifying only the profile ID and (if required) PIN.

This example activates profile 12345, which has PIN abc1234, and leaves the user to complete configuration steps manually.

iOS

```
ipass://provision?pid=12345&pin=abc1234
```

Android

```
clientx://provision.com?pid=12345&pin=abc1234
```

Reconfiguration

If your organization implements security policies such as periodic password change, user devices can be transparently reconfigured without the participation of the user. A simple call to the provisioning API, specifying the device owner's password, will cause it to be replaced.

This example changes the password for the device user to "snookums12"

iOS

```
ipass://provision?password=snookums12
```

Android

```
clientx://provision.com?password=snookums12
```

Reactivation

Reactivation changes the Open Mobile profile in use by an active device. Reactivation could be required if you, for example, decide to make higher-cost networks available to just a small subset of your existing user population.

To reactivate a device and enable the reactivated used to access these new networks, create or select a new profile on the Open Mobile Portal, assign the additional networks for use with the profile, and then selectively reactivate the targeted users by performing a simple activation request with the provisioning API. As long as you do not change domains or prefixes, after reactivation, the user population will be able to access new networks enabled by the profile, but will not have to set username or password, or select domain or prefix.

This example re-provisions a device with profile 13144, which has no PIN.

iOS

```
ipass://provision?pid=13144
```

Android

```
clientx://provision.com?pid=13144
```

Reprovisioning: New Device User

When a device is assigned to a new user, you may need to reprovision the device with a profile and settings appropriate for the new user. Create a “one-shot” call to the provisioning API, very similar to the initial provisioning example, above.

iOS

```
ipass://provision?pid=9945&username=SteveMoss&password=pinkkitty3&domain=mycorp.com
```

Android

```
clientx://provision.com?pid=9945&username=SteveMoss&password=pinkkitty3&domain=mycorp.c
om
```

Silent Provisioning: New Device User

This example is a “silent” version of the one-shot Complete Provisioning discussed early. Users will only get a notification indication the result of provisioning, without seeing Open Mobile brought to foreground at all.

Android

```
clientx://provision.com?pid=9945&username=jsmalley&password=doggylove2&domain=mycorp.co
m&silentmode=Y
```

iOS

```
iPass://provision.com?pid=9945&username=jsmalley&password=doggylove2&domain=mycorp.com&
silentmode=Y
```

Password Obfuscation: New Device User

The password field in this example is obfuscated. Note that obfuscation flage needs to be Y so that the app knows unobfuscation is required.

Android

```
clientx://provision.com?pid=9945&username=jsmalley&password=AEB1DB802DF810239ECA9013
&domain=mycorp.com&obfuscation=Y
```

iOS

```
iPass://provision.com?pid=9945&username=jsmalley&password=AEB1DB802DF810239ECA9013
&domain=mycorp.com&obfuscation=Y
```