Using Direct Update to quickly update your application

Overview

With Direct Update, Cordova applications can be updated "over-the-air" with refreshed web resources.

Benefits of Direct Update:

- Using Direct Update, organizations can ensure that users always use the latest version of the application.
- Application versions are better controlled. Users are notified of pending updates or prevented from using obsolete versions.
- Updates that are deployed to the MobileFirst Server are automatically pushed to user devices.

Restrictions:

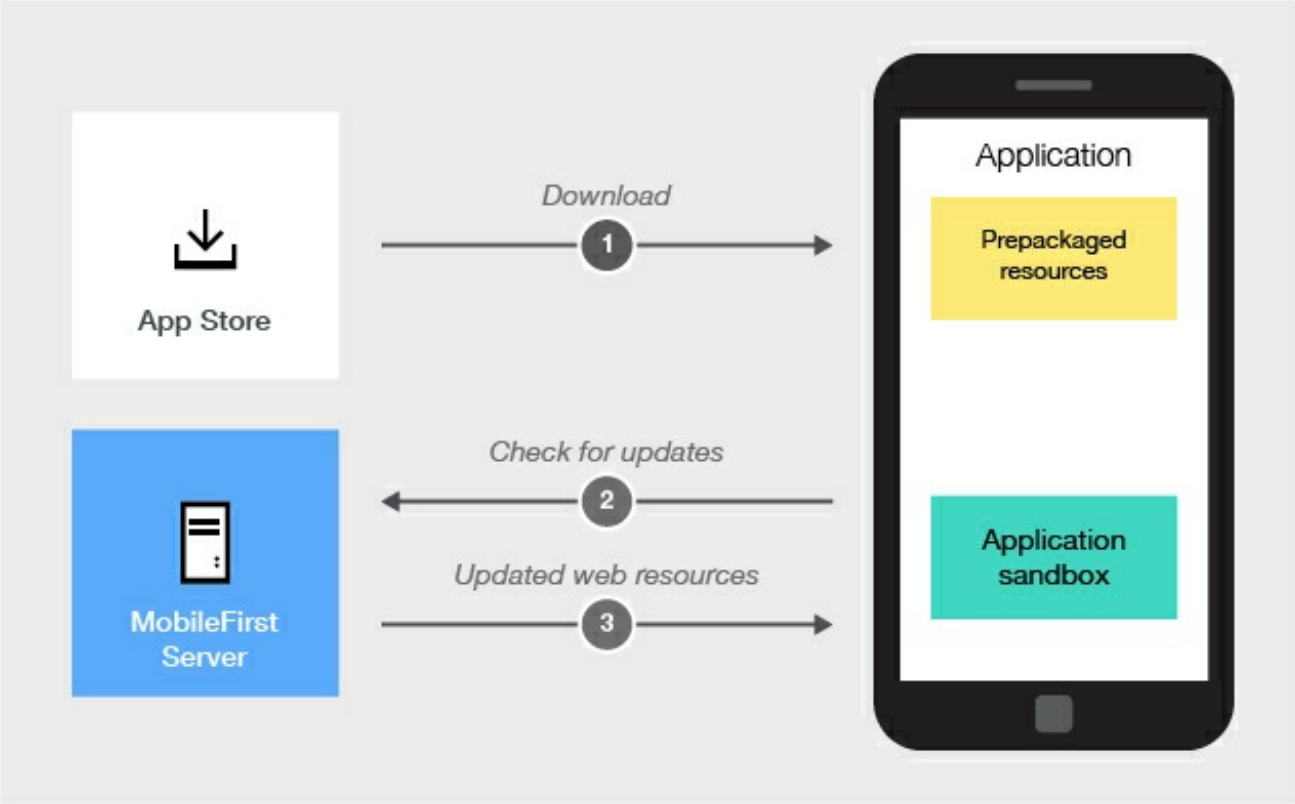
- Direct Update updates the app web resources only.
- To update native resources a new app version must be uploaded to the respective app store.
- Android: no restrictions.
- Windows 8 and Windows 10: no restrictions.
- iOS:
    - B2C: according to the terms of service of your company; usually at least bug fixes are allowed.
    - B2E: through the iOS Developer Enterprise Program.

Jump to:

How Direct Update works

The application web resources are initially packaged with the application to ensure first offline availability. Afterwards, the application checks for updates based on its configuration. The updated web resources are downloaded when necessary.

After a Direct Update, the application no longer uses the pre-packaged web resources. Instead it will use the web resources in the application's sandbox.

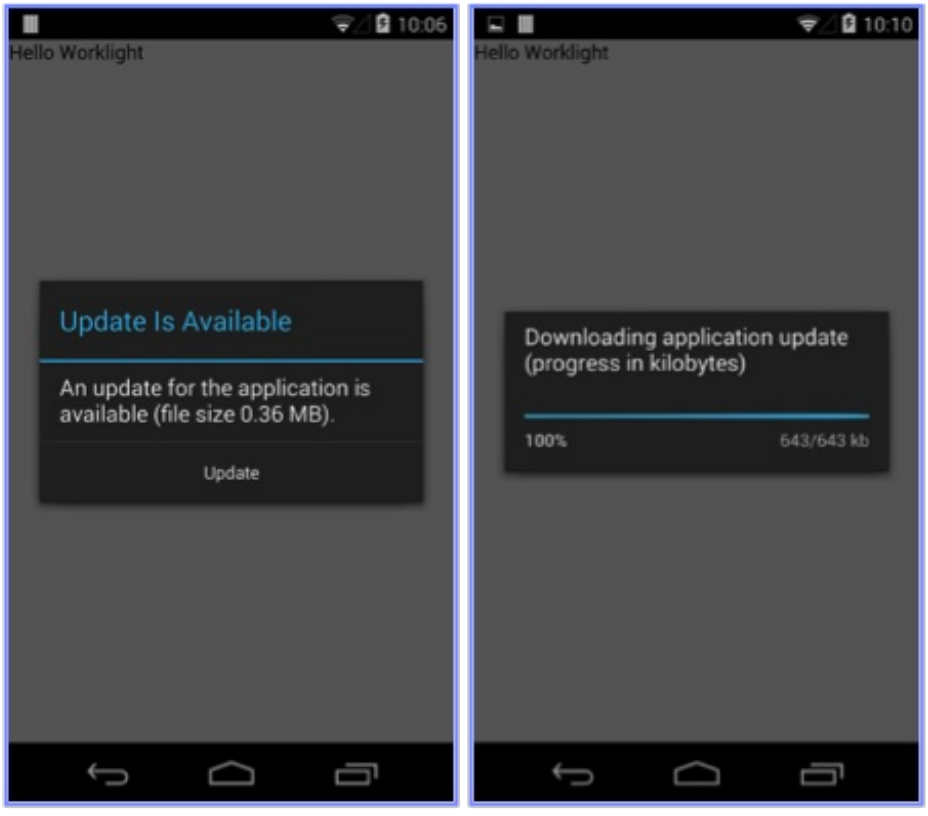


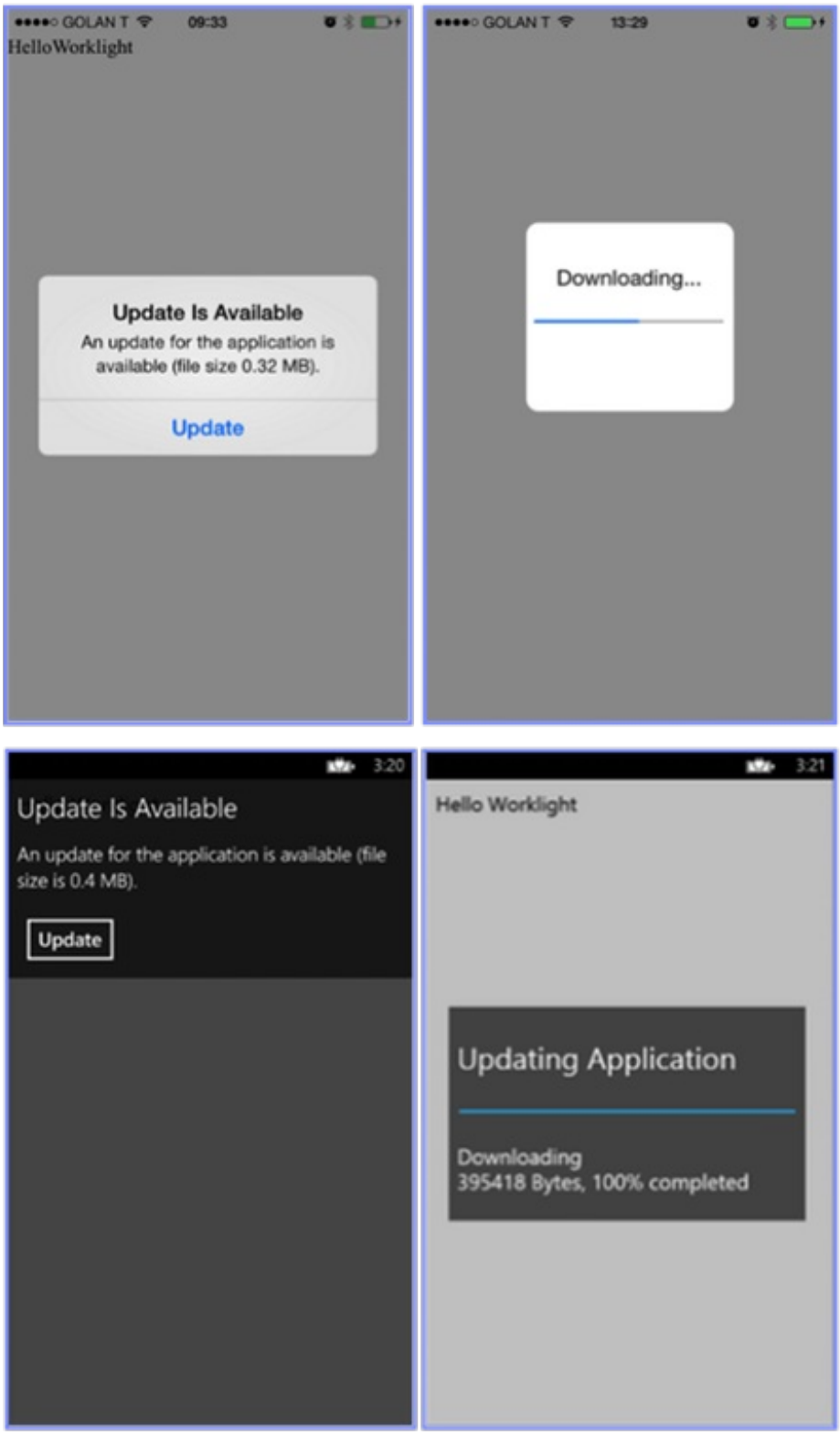
Deploying updated web resources to MobileFirst Server

TODO: how to create updated web resources and upload to the console

User Experience

By default, after a Direct Update is received a dialog is displayed and the user is asked whether to begin the update process. After the user approves a progress bar dialog is displayed and the web resources are downloaded. The application is automatically reloaded after the update is complete.





Customizing the Direct Update UI

It is possible to override the default UI and/or UX of Direct Update, and create a custom Direct Update behavior altogether.
To do so, override the `handleDirectUpdate` function:

```
wl_DirectUpdateChallengeHandler.handleDirectUpdate = function(directUpdateData,
directUpdateContext) {
    // custom Direct Update logic
};
```

- `directUpdateData` - A JSON object containing the `downloadSize` property that represents the file size (in bytes) of the update package to be downloaded from MobileFirst Server.
- `directUpdateContext` - A JavaScript object exposing the `.start()` and `.stop()` functions, which start and stop the Direct Update flow.

Example

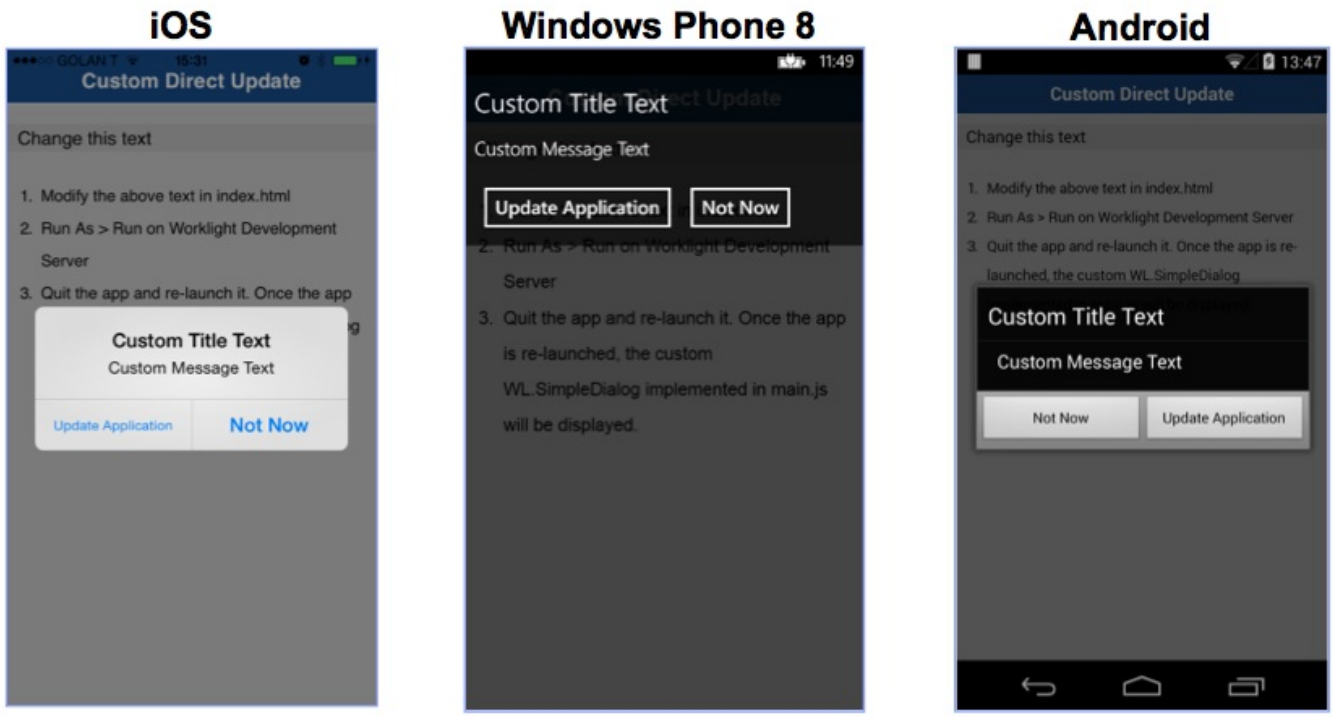
In the example code below, a custom Direct Update dialog is presented for the user to either continue with the update process or dismiss it.
Additional examples for customized Direct Update UI:

- A dialog that is created by using a third-party JavaScript framework (such as Dojo or jQuery Mobile, Ionic, ...)
- Fully native UI by executing a Cordova plug-in
- An alternate HTML file that is presented to the user with options
- And so on…

```
wl_directUpdateChallengeHandler.handleDirectUpdate = function(directUpdateData,
directUpdateContext) {
    // custom text for the dialog
    var customDialogTitle = 'Custom Title Text';
    var customDialogMessage = 'Custom Message Text';
    var customButtonText1 = 'Update Application';
    var customButtonText2 = 'Not Now';

    // Create dialog
    navigator.notification.confirm(
        'Custom Message Text',
         onClick,
        'Custom Title Text',
        ['Update','Cancel']
    );
};

// Handle dialog buttons
function onClick(buttonIndex) {
    if (buttonIndex == 1) {
        directUpdateContext.start();
    } else {
        wl_directUpdateChallengeHandler.submitFailure();
    }
}
```



In the example above, the `submitFailure` API is used to dismiss the Direct Update:

```
wl_directUpdateChallengeHandler.submitFailure();
```

As mentioned, when the developer creates a customized Direct Update experience, the responsibility for its flow now belongs to the developer.
As such, it is important to call `submitFailure()` to notify the MobileFirst framework that the process completed with a "failure". The MobileFirst framework in turn invokes the `onFailure` callback of the invocation that triggered the Direct Update.

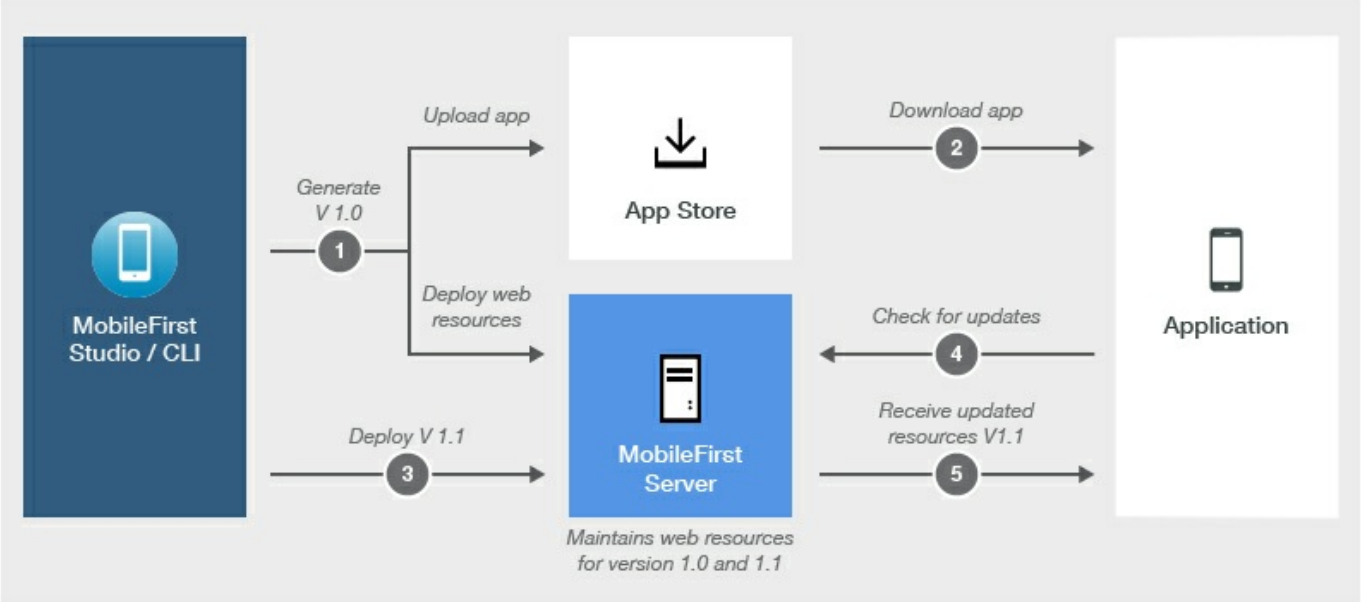
Because the update process did not take place, it will occur again the next time it is triggered. Optionally, a developer can also supply a Direct Update listener to fully control a Direct Update lifecycle.

```
directUpdateContext.start(directUpdateCustomListener);
```

> For more information, see the topic about customizing the direct update interface, in the user documentation.

Working with Direct Update in the field

The diagram below depicts the flow of updating an application's web resources using Direct Update once it has been submitted to the application stores and used by end-users.

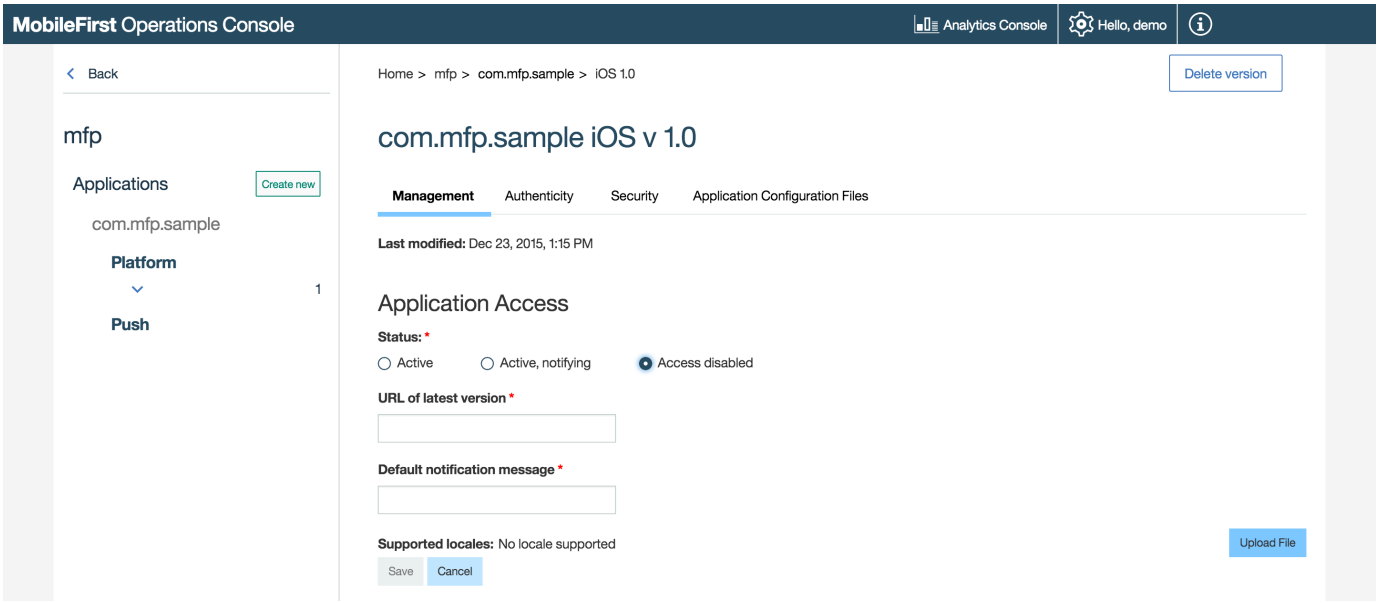


Note: During development cycles, testers automatically get recent web resources through internal distribution mechanisms and not through application stores.

Disabling old application versions

From the MobileFirst Operations Console, it is possible to prevent users from using obsolete versions, and to notify users about available updates.

Clarification: The Remote Disable feature only prevents users from interacting with MobileFirst Server; that is, it prevents the app from connecting to the server. The application itself is still accessible. Any action in the application that requires server connectivity is blocked.



Direct Update authenticity

Direct Update authenticity prevents a 3rd-party attacker from altering the transmitted web resources from the server (or if it is stored at a content delivery network (CDN))to the client application.

Direct Update authenticity is always-on and uses the certificate stored in the application server's keystore.

Notes:

- Direct Update authenticity does not work on already-deployed applications. Applications must be re-deployed to the various app stores.

Differential Direct Update

Differential Direct Updates enables an application to download only the files that were changed since the last update instead of the entire web resources of the application. This reduces download time, conserves bandwidth, and improves overall user experience.

Important: A differential update is possible only if the client application's web resources are one version behind the application that is currently deployed on the server. Client applications that are more than one version behind the currently deployed application (meaning the application was deployed to the server at least twice since the client application was updated), receive a full update - meaning that the entire web resources are downloaded and updated.

There is no change in the behaviour of applications that were built with previous versions of IBM MobileFirst Platform Foundation.