AlertFlag (version 2.0.0)

Developer Documentation

Workflow Data Systems LLC
Code: James David Ramsey
Graphics: Harold Leber
Custom Sounds: jkhinzman

Table of Contents

Caveats

This documentation is a work-in-progress. If you find anything amiss, or can't find something that you really need, please let us know.

The Story of AlertFlag

At FileMaker DevCon 2012, I saw something amazing. Rich Carlton demonstrated a proof-of-concept iOS app called FM Notify that would give FileMaker Go developers the ability to schedule OS-level notifications from within FileMaker Go. Even if the app had been quit, even if FileMaker Go had been quit, the notifications would still appear. It required the installation of a separate application on the user's device, but after that it very nearly disappeared. A flash or two on-screen was all the user might experience.

And they'd successfully gotten it approved.

After the session, I told Rich how important I felt this was for the community, and how grateful I was that they had not only developed this tool, but that they had provided it free of charge. Yes, and then I had a stack of feature requests.

I told Rich I felt the existence of this app was so critical to the community that if he ever decided he didn't want to continue to support the app, that I would be happy to maintain it.

Fast forward a month, and I get an email from Rich that Richard Carlton Consulting is *really* busy, and they're not going to have the resources to spend on maintaining the app… and would I like to move it forward?

Fast forward 5 months, and version 2 is ready for release. I took the soul of the concept that Rich demoed, and pounded on it until I had a tool that was fast, scalable, flexible, and had room to grow. I hope you find that it meets your needs. If not, please let me know. We'll see what we can do.

James David Ramsey
dave@workflowdata.com

New Style URL

AlertFlag version 2 uses a new URL style to create notifications. The new system uses named parameters to increase readability, and to allow for parameters to appear in any order without impacting functionality. Long-term, this will also make it easier to add new parameters without breaking everyone's existing solutions.

Here's a basic sample:

```
alertflag://&appname=Events&callback=fmp:%2F%2F~%2FFM%2BNotify.fmp12
      %3Fscript%3DFM_Notify_Response_Handler&message=This%20will%20p
      op%20up%20later&timeoffset=30
```

If your URL's readability will be aided by seeing the offset first, this other URL will do the same thing:

```
alertflag://&timeoffset=30&appname=Events&callback=fmp:%2F%2F~%2FFM%
      2BNotify.fmp12%3Fscript%3DFM_Notify_Response_Handler&message=T
      his%20will%20pop%20up%20later
```

All we did was move the `&timeoffset` parameter to the beginning, rather than the end.

For readability, we will add linefeeds to all future example URLs. These line feeds should be removed if you want to use the URL. For example, the third URL will be written like this:

```
 alertflag://
 &timeoffset=30
 &appname=Events
 &callback=fmp:%2F%2F~%2FFM%2BNotify.fmp12%3Fscript%3DFM_Notify_Resp
      onse_Handler
 &message=This%20will%20pop%20up%20later
```

This should make it a little easier to see which parameters have been included, and to see the values of those parameters

General Parameter Guidelines

Any parameter which contain a URL, a message, an app name, button text, etc should be properly encoded to be included in a URL. In FileMaker Pro, this text should be encoded using the `GetAsURLEncoded()` calculation function. In the following detailed description of available parameters, any parameter which needs to be encoded will include the [ENCODE] indicator

Some parameters have a specific list of permitted values. For example, the "&repeating" parameter can have only the following values:

```
off
daily
weekly
hourly
```

In the description of this parameter, these options will be expressed as follows: [OPTIONS: off/daily/weekly/hourly]

All parameters fall into one of three categories: REQUIRED, SEMI-REQUIRED, and OPTIONAL.

REQUIRED

Any parameter marked as REQUIRED /must/ be included in any create request sent to AlertFlag. Failure to include one of these will generate an error. Please note that it is possible for an error to be so significant that AlertFlag either never receives the URL, or is incapable of understanding enough of the URL to generate a meaningful error. More on this in the section 'Debugging'.

SEMI-REQUIRED

Parameters marked as SEMI-REQUIRED are clustered into groups While no particular parameters in the group is required, at least one item from the group is required in every creation request. For example one SEMI-REQUIRED group contains `&timestamp` and `&timeoffset`. These parameters control when the notification will appear. `&timestamp` designates a particular date and time that the notification should occur. `&timeoffset` designates a number of seconds that AlertFlag should wait before displaying the notification. Without one of these values, AlertFlag would have no idea when the notification should be displayed, so you have to provide one or the other in every creation request. An error will be generated if one of these parameters is not provided. In this particular case, an error message is also created if *both* are provided, as a conflict would be generated.

OPTIONAL

Parameters designated as optional are only needed if you want to override their default values. For example, the `&button` parameter controls the text on the non-cancel button if the notification is displayed as a dialog (rather than a banner.... the use controls this in their notification settings). The default value of this button is "View" (in English... As an added bonus, iOS will automatically localize this default value for the current language). If you would like to keep this default value, simply don't provide this parameter. If, however, you would like to display a different message on this button, you should provide new button text using the `&button` parameter.

The Parameters

APPNAME - REQUIRED - [ENCODE]

The `&appname` parameter is used by AlertFlag's interface, to label the notification by the application that generated it. While you could simply say 'FileMaker'

```
&appname=FileMaker
```

or 'FileMaker Go'

```
&appname=FileMaker%20Go
```

There's a decent chance that your users would prefer to see the name of *your* application

```
&appname=SeedCode%20Calendar
```

This value is also passed back when the `&style` parameter (described below) equals `filemaker` as a local variable in the callback.

MESSAGE - REQUIRED - [ENCODE]

The `&message` parameter is used to designate the text that you would like to see displayed in the notification box (or notification banner) when the notification is displayed to the user. Your message should suggest to the user why they would want to click on this notification, so you will likely want to avoid messages like "Click Here", or "Reminder". At the same time, you are limited in the amount of text that can be displayed here, so be brief. How about "It is time to complete your daily activity report."

```
&message=It%20is%20time%20to%20complete%20your%20daily%20activity%20report.
```

SEMI-REQUIRED GROUP #1 - Time

Every notification needs to have a designated time when it should appear. There are two possible ways to designate this: `&timestamp` and `&timeoffset`. One of these two must exist in every notification creation URL. Each is explained in detail below. Please note that while one of the parameters is required, only one per notification is permitted. If you proved both `&timestamp` and `&timeoffset` for a single notification, no notification is scheduled, and an error is generated.

TIMESTAMP - SEMI-REQUIRED

The `&timestamp` parameter is looking for a date and time designated by the number of seconds since Midnight, January 1st, 1970, GMT. For FileMaker developers, this is a different baseline time than FileMaker uses for its timestamps (January 1st, 0001). So, to generate an AlertFlag timestamp from a FileMaker timestamp, use `getAsNumber()` and then subtract 62135596800 (which is the number of seconds between these two reference dates). So:

```
&timestamp=1423040640
```

indicates that the notification should go off on February 4, 2015 at 9:04:00 AM.

TIMEOFFSET - SEMI-REQUIRED

Using the `&timeoffset` parameter indicates that the notification should be displayed the provided number of seconds in the future. So

```
&timeoffset=3600
```

indicates that the notification should be displayed exactly one hour from now. Alternately,

```
&timeoffset=86400
```

indicates that the notification should be displayed exactly 24 hours after it is created.

Time Zones

All notifications are created as if they will go off in the same time zone in which they were scheduled. If you have users that operate in multiple timezones, you may have to make adjustments. For example, if a user schedules a notification to go off at 9:00 AM tomorrow morning while they are in the Eastern Time Zone, and they then get on a plane and fly to California, the notification will go off at 6:00 AM the following morning. If, however, they schedule the notification to go off at 9:00 AM while they are in California, and then fly to New York, the notification will go off at noon.

SEMI-REQUIRED GROUP #2 - Callbacks

When the user clicks on the notification, AlertFlag needs a URL to open to pass control to the appropriate application. A properly scheduled notification needs to know how to handle 4 events:

Creation - After the notification has been scheduled, what URL is used pass control back to the calling application?

Deletion - After the notification has been deleted, what URL is used pass control back to the calling application?

Error - If an error was generated with a particular attempted creation of a notification, what URL is used pass control back to the calling application (along with details of the error)?

Popping - When the notification is displayed to the user, and they click "View" (or whatever the developer has designated), what URL is launched?

In terms of FileMaker, there are a couple ways to handle this. Since the parameters passed back with any URL include what event has occurred, you can construct a single script that knows how to handle any of the above 4 events. Then, you really just need to pass one URL to AlertFlag. For each kind of event, AlertFlag will append parameters describing what happened. In this case, you would use the `&callback` parameter.

However, you also have the option of providing an alternate URL for any of these 4 events. So, if you wanted a different URL to be returned in the case of an error, you could provide a different URL using the `&callbackcancel` parameter. If you wanted a different URL to be launched when the notification is popped, you could provide that using the `&callbackpopped` parameter.

Since a valid notification needs to know how to handle all 4 events, you either need to provide URLS for ***all four*** specific callbacks (`&callbackcreation`, `&callbackpopped`, `&callbackcancel`, and `&callbackcreationerror`), or you must provide a failover callback using the `&callback` parameter. Alternately, If you provide the `&callback` parameter, you may provide any combination of the four specific callbacks, including the option of providing *none* of them.

If an event occurs, the AlertFlag checks for one of the event-specific callbacks first. If it doesn't find one, it will use `&callback` instead.

UUID - REQUIRED (but only when &action=cancel)

The `&uuid` parameter is used to designate a specific notification to cancel. When a notification is created using the FileMaker style, the return callback includes the UUID of the notification as a local variable that can be accessed from your callback script. This parameter is currently ignored in all other styles.

STYLE - OPTIONAL

The `&style` parameter is used to customize how AlertFlag interacts with the callback URLs. This allows AlertFlag to offer optional extensions to those URLs. If `&style=filemaker` (or is left blank, as `filemaker` is the default value), four local variable declarations are added to the callback URL:

`$ALERTFLAG_Action` - [Possible Values: `create` / `cancel` / `popped` / `creationerror`]

`$ALERTFLAG_Identifier` - The value passed to AlertFlag with the `&identifier` parameter.

`$ALERTFLAG_UUID` - the UUID of the notification

`$ALERTFLAG_ErrorFlag` - a `1` if an error occurred, an empty string ("") if no error occurred.

`$ALERTFLAG_ErrorMessage` - Human readable error messages from AlertFlag. In some cases, multiple error messages will be returned. These error messages will be separated by a forward slash

`$ALERTFLAG_AppName` - the value originally passed as `&appname` is returned in this local variable.

IDENTIFIER - OPTIONAL

The `&identifier` parameter is used to pass a piece of FileMaker specific information that will be passed back as a parameter with the callback (for example: a primary key, a date, a user name, etc). The value of the identifier is passed back to FileMaker as the local script variable `$ALERTFLAG_Identifier`.

The default value for `&identifier` is blank.

This parameter currently only is used if the `&style` is `filemaker` (or undesignated).

ACTION - OPTIONAL [OPTIONS: create/cancel]

The parameter `&action` is used to indicate to AlertFlag if it is creating a new notification or canceling an existing notification. There is no action for editing / rescheduling an existing notification. If you need to edit or otherwise reschedule an existing notification, you should cancel that notification and create a new one in its place.
If the `&action` is undesignated, the default value is create.
`&action=create` indicates that AlertFlag should create a new notification using the other parameters that were included in this URL.
`&action=cancel` instructs AlertFlag to cancel an existing notification. The only other parameter that is required (or even used) is the `&uuid` parameter. For further information, please see the section on canceling notifications.

REPEATING - OPTIONAL [OPTIONS: off/daily/weekly/hourly]

The parameter `&repeating` controls whether the notification is discarded when clicked on by a user, or if a repetition is rescheduled for a later time. If this parameter is omitted, the notification will not repeat at all. This is equivalent to explicitly designating &repeat=off.

`&repeat=daily` repeats 24 hours after the original scheduled notification time.
`&repeat=weekly` repeats 7 days after the original scheduled notification time.
`&repeat=hourly` repeats 1 hour after the original scheduled notification time.

(for testing purposes only)
`&repeat=minutely` repeats every 60 seconds after the original scheduled notification time. DO NOT USE THIS VALUE in a live production system. This should only be used if you need to test the repetition behavior on a shorter timeline than once every hour.
(for testing purposes only)

RECURRENCE

if `&repeat=daily` is used, the notification (once acknowledged) will be rescheduled to repeat (with the same IDs, same message, same callbacks, same UUID, etc) for 24 hours after the original scheduled notification time. It is NOT dependent upon the time that the notification is acknowledged. So, if the notification is set to go off at 2:00 PM, but the user does not click on the notification until 4:00 PM, the next notification will go off at 2:00 PM the next day.

If the user does not click our theoretical 2:00 PM notification (and there are no other scheduled notifications) the notification will not repeat unless AlertFlag is activated in some way. If AlertFlag is never ever reactivated, the notification will never repeat.

However, if AlertFlag is reactivated for any reason (via URL or via manual user interaction), any scheduled repeating notifications will be rescheduled in line with their repetition rate and their originally scheduled time.

There is not (at present) any way to schedule a truly repeating notification in iOS without some kind of application activity. If you need to get as close as possible, consider scheduling 7 notifications for 7 successive days to repeat *weekly* rather than daily. The "repetition" would only be stopped if the user ignored all 7 notifications (and AlertFlag was not otherwise activated during that week).

BUTTON - OPTIONAL [ENCODE]

The `&button` parameter is used to take manual control of the button that is displayed with the notification dialog, assuming that the user has set AlertFlag to display its notifications in a dialog, rather than as a banner. If the notification is displayed as a banner, this parameter is used in the "slide to unlock" bar. `&button=Retrieve` changes the slider message to "slide to retrieve". Note that iOS forces the value to lowercase.
If `&button` is not provided, the default behavior is for iOS to display the button with a localized version of the word "view", based upon the default language of the user. Please note that this default behavior is different than `&button=view`, as this explicit designation is not localized. It will display the word "view" regardless of the preferred language of the user.
Please note that there is limited screen real estate available for this button label. Try not to get too verbose.

SOUND - OPTIONAL [OPTIONS: default/silent/alertflagBell]

The `&sound` parameter allows the developer to specify the noise emitted when the notification is displayed. Please note that no sound will be emitted (regardless of the value of this parameter) in the following cases:

--if the user has set AlertFlag notifications to not generate noise (in the Notifications section of the Settings app)

--if their device is set to Do Not Disturb,

--if they have turned off the speaker for their device

If this parameter is not included in the creation URL, AlertFlag will attempt to use the default notification sound as designated by the user in the Settings app. To designate this explicitly, use `&sound=default`.

`&sound=silent` instructs AlertFlag to make no noise when the notification is displayed, regardless of the user-settings. This is particularly useful when the notification to be displayed is not exceptionally time-sensitive, and it will be acceptable if the user does not see the notification until the next time they unlock their device.

`&sound=alertflagBell` tells AlertFlag to override the default notification sound, and to instead use a custom tone created specifically for AlertFlag. This is best when it would helpful for the user to be able to distinguish AlertFlag notifications from those of other applications based upon sound alone.

ERRORMESSAGE - OPTIONAL [DEFAULT:no] [OPTIONS: no/yes]

The parameter `&errormessage` controls whether or not AlertFlag displays a dialog with the error messages when an error is encountered while creating a notification. This should only be set to `yes` while the developer is working out new code.

When working on a new notification, if an error of sufficient size is encountered, then no return callback will be generated. You will be unable to pull the error messages out of the `$ALERTFLAG_ErrorMessage` variable when the callback occurs, because it either *didn't* occur, or failed to properly interface with your FileMaker-side script. When debugging issues of this scale, it is helpful to be able to see the error message anyway. This is also helpful when working with non-FileMaker `&style` values, as the error codes are not appended to the callback.

Please note that if the callback *does* fire properly, AlertFlag will be sent to the background as normal. Setting `&errorMessage=yes` will not interrupt the functioning of the process or callbacks. In his case, the dialog will only be visible if AlertFlag is re-activated.

Please note that it is still possible to generate an error of such a scale that no dialog will be displayed. AlertFlag makes a good-faith effort to parse your URL, but a poorly formed URL can still completely circumvent all the protections put in place.

Sample URLs

Below are some more sample URLs

For now, please visit http://workflowdata.com/alertflag.html for the latest samples

[STILL TO COME…]

Sample FileMaker Applications

There are some sample applications available…

For now, please visit http://workflowdata.com/alertflag.html for the latest samples

[STILL TO COME…]

Debugging

There are some sample applications available…

For now, please visit http://workflowdata.com/alertflag.html for the latest samples

[STILL TO COME…]