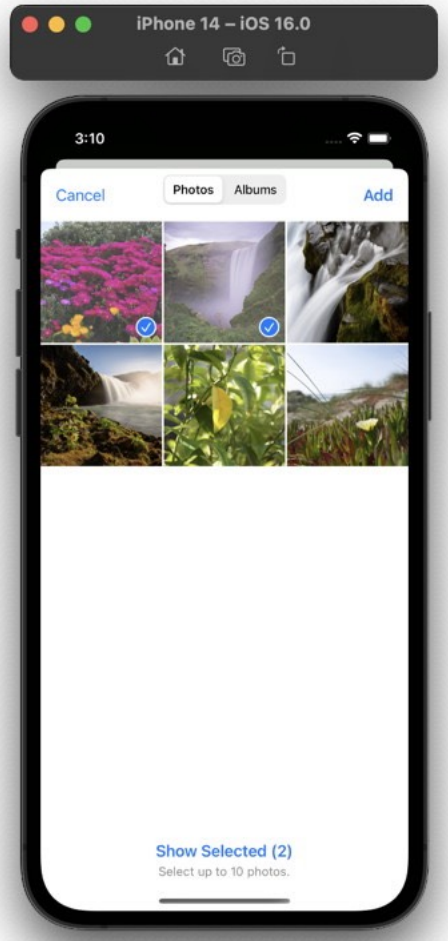
PhotoPicker for iOS with FileMaker iOS SDK



Next to your container fields in iOS, you can show a button to insert a picture from the photo libraries. If you'd do this with our older ImagePicker functions, you may get a dialog about permissions to access the photos library.

The new PhotoPicker functions allow you to access a dialog to pick photos from your app without such a dialog. The dialog runs as part of a background app included with iOS, which has all the permissions needed. It can show the whole photo library without your application having access to it. The user can select one or more pictures and have those being available to your applications. The images are then downloaded in the background if needed and you can check progress in a script.

Here is a sample script:

```
If [ MBS( "PhotoPicker.Available") ]
        Set Variable [ $r ; Value:
MBS( "PhotoPicker.Clear") ]
        # set trigger
        Set Variable [ $r ; Value:
MBS( "PhotoPicker.SetScript"; Get(FileName);
"GotImage") ]
        # you can define a limit, 0 = no limit
        Set Variable [ $r ; Value: MBS( "PhotoPicker.SetSelectionLimit"; 10) ]
        # show the picker!
        Set Variable [ $r ; Value: MBS( "PhotoPicker.Present") ]
End If
```

And the script triggered looks like the one below. Basically when the script is triggered, we look how many bytes have to be loaded and show a progress dialog for this. We wait until all images are downloaded. Then we loop over the images and read each one into a container field. For HEIC images use our our Container.ReadImageFile function since that function converts the picture then to PNG format. The clear function later removes the images from memory when we don't need them any longer. Take a look on this script:

```
Set Variable [ $r ; Value: MBS("ProgressDialog.SetTopText"; "Loading images") ]
Set Variable [ $r ; Value: MBS("ProgressDialog.SetProgress"; -1) ]
Set Variable [ $r ; Value: MBS("ProgressDialog.Show") ]
Pause/Resume Script [ Duration (seconds): ,5 ]
Loop
	Set Variable [ $BytesLoaded ; Value: MBS("PhotoPicker.BytesLoaded") ]
	Set Variable [ $TotalBytes ; Value: MBS("PhotoPicker.TotalBytes") ]
	Set Variable [ $r ; Value: MBS("ProgressDialog.SetProgress"; $BytesLoaded
* 100 / $TotalBytes ) ]
	#
	If [ MBS("PhotoPicker.IsLoading") ]
		Pause/Resume Script [ Duration (seconds): ,1 ]
	Else
		Exit Loop If [ 1 ]
	End If
End Loop
Set Variable [ $r ; Value: MBS("ProgressDialog.Hide") ]
#
Set Variable [ $ImageCount ; Value: MBS("PhotoPicker.ImageCount") ]
Set Variable [ $index ; Value: 0 ]
If [ $index < $ImageCount ]
	Loop
		# Process each image
		Set Variable [ $FilePath ; Value: MBS("PhotoPicker.File"; $index) ]
		Set Variable [ $Error ; Value: MBS("PhotoPicker.Error"; $index) ]
		#
		If [ Length ( $FilePath ) > 0 ]
			New Record/Request
			If [ Right ( $FilePath; 5 ) = ".heic" ]
				# special case for HEIC images -> we convert them to
PNG
				Set Variable [ $Image ; Value:
MBS( "Container.ReadImageFile"; $FilePath; "PNG"; "image.png") ]
			Else
				# all other files get imported
				Set Variable [ $Image ; Value:
MBS( "Container.ReadFile"; $FilePath) ]
			End If
			Set Field [ ImagePicker::Picture ; $image ]
			Commit Records/Requests [ With dialog: Off ]
		End If
		#
		# next
		Set Variable [ $index ; Value: $index + 1 ]
		Exit Loop If [ $index ≥ $ImageCount ]
```

```
        End Loop
End If
Set Variable [ $Image ; Value: MBS( "PhotoPicker.Clear") ]
```

Please try the new functions in MBS Plugin 12.5 or newer.