Read and write notes in Contacts

As you may know we have Contacts functions for years and quite a few people use them to synchronize to the macOS Contacts application from FileMaker. Our functions allow you to read and write to contacts database and you can implement a sync method for it. Especially if you have contacts in your FileMaker database, you may like to synchronize these contacts to the contacts database. Once synchronized, they appear on the iPhone when you get called.

We got a bit surprised with macOS Ventura adding restrictions for reading notes. Don't get us wrong: the privacy control is great. Like you may not like every application to read your whole contacts database. Especially as people write sensitive informations in their notes, this field gets an extra permission. This allows you to share your contacts with an application like Signal or WhatsApp, but not upload all these notes. Now FileMaker Pro comes with the entitlement to read the contacts, but not the one for the note field. So we looked for alternative ways and found one via AppleScript.

Since we have AppleScript functions in our plugin to run AppleScripts, we can leverage them. In our case we have a static script with a property, which we compile once. Then we can use it several times and just change the ID we pass. Please never build scripts using data from the user. We don't want the equivalent of a SQL injection in our AppleScript script. For each time you lookup the contact note, you can now take AppleScript.SetPropertyValue to pass the ID and the run AppleScript.Execute function. This then returns the result of the script, which is the note field.

Here is the sample script:

```
# Read Contact Note via AppleScript
Set Variable [ $theID ; Value: "773E1A42-ABCD-4CCF-88DE-
E9C64BEE6D33:ABPerson" ]
# Compile AppleScript
Set Variable [ $ScriptID ; Value: MBS( "AppleScript.Compile";
"property theID : \"\"¶
tell application \"Contacts\"¶
        set theContact to the first person whose id is theID¶
        return note of theContact¶
end tell" ) ]
# run a query or multiple
Set Variable [ $r ; Value: MBS( "AppleScript.SetPropertyValue"; $ScriptID; "theID";
$theID ) ]
Set Variable [ $result ; Value: MBS( "AppleScript.Execute"; $ScriptID) ]
```

```
Show Custom Dialog [ "Result from AppleScript" ; $result & ¶ &
MBS( "AppleScript.LastError" ) & ¶ & MBS( "AppleScript.LastErrorMessage" ) ]
# free memory
Set Variable [ $r ; Value: MBS( "AppleScript.Close"; $ScriptID) ]
```

The same way can be used to write the note field. We make a script with two properties, one for the ID and one for the note. Then we compile it and whenever we need to save a note, we can just use AppleScript.SetPropertyValue to fill in the two parameters. Then execute will run and set it. The script calls save to write changes back to the contacts database and then returns the new value. Be sure to check for error status.

Our write script:

```
# Set Contact Note via AppleScript
Set Variable [ $theID ; Value: "773E1A42-ABCD-4CCF-88DE-
E9C64BEE6D33:ABPerson" ]
Set Variable [ $theNote ; Value: "Best Hotdogs in town!" ]
# Compile AppleScript
Set Variable [ $ScriptID ; Value: MBS( "AppleScript.Compile";
"property theID : \"\"¶
property theNote : \"\"¶
tell application \"Contacts\"¶
        set theContact to the first person whose id is theID¶
        set note of theContact to theNote¶
        save¶
        return note of theContact¶
end tell" ) ]
Show Custom Dialog [ "Result from AppleScript" ; MBS( "AppleScript.LastError" )
& ¶ & MBS( "AppleScript.LastErrorMessage" ) ]
# run a query or multiple
Set Variable [ $r ; Value: MBS( "AppleScript.SetPropertyValue"; $ScriptID; "theID";
$theID ) ]
Set Variable [ $r ; Value: MBS( "AppleScript.SetPropertyValue"; $ScriptID;
"theNote"; $theNote ) ]
Set Variable [ $result ; Value: MBS( "AppleScript.Execute"; $ScriptID) ]
Show Custom Dialog [ "Result from AppleScript" ; $result & ¶ &
MBS( "AppleScript.LastError" ) & ¶ & MBS( "AppleScript.LastErrorMessage" ) ]
# free memory
Set Variable [ $r ; Value: MBS( "AppleScript.Close"; $ScriptID) ]
```

When you run these scripts, you may get a question from macOS about whether FileMaker is allowed to control the Contacts application. You can use Applescript.DeterminePermissionToAutomateTarget function to ask for permissions before front. You can check for the status or show the dialog

Enjoy setting note field and otherwise use the regular contacts functions in our plugin as they are faster to do read and write data.