

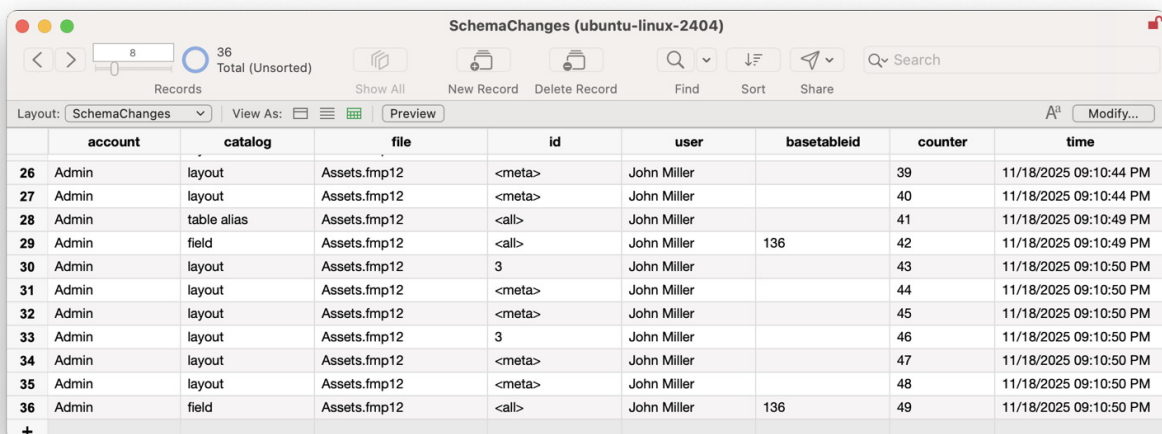
Watching for schema changes in your FileMaker Server

Do you have a FileMaker Server with multiple databases? Do you like to keep track who makes changes?

With FileMaker Server 21.1 the Plugin SDK got a new entry point for requesting schema changes. This provides a way for the plugin to be notified about changes to any of the databases on the server. With changes to the schema, we mean changes for these catalogs:

- custom function
- custom menu set
- data source
- extended privilege
- field
- layout
- privilege set
- script
- table
- table alias
- theme
- user account
- value list

There may be more catalogs.



	account	catalog	file	id	user	basetableid	counter	time
26	Admin	layout	Assets.fmp12	<meta>	John Miller		39	11/18/2025 09:10:44 PM
27	Admin	layout	Assets.fmp12	<meta>	John Miller		40	11/18/2025 09:10:44 PM
28	Admin	table alias	Assets.fmp12	<all>	John Miller		41	11/18/2025 09:10:49 PM
29	Admin	field	Assets.fmp12	<all>	John Miller	136	42	11/18/2025 09:10:49 PM
30	Admin	layout	Assets.fmp12	3	John Miller		43	11/18/2025 09:10:50 PM
31	Admin	layout	Assets.fmp12	<meta>	John Miller		44	11/18/2025 09:10:50 PM
32	Admin	layout	Assets.fmp12	<meta>	John Miller		45	11/18/2025 09:10:50 PM
33	Admin	layout	Assets.fmp12	3	John Miller		46	11/18/2025 09:10:50 PM
34	Admin	layout	Assets.fmp12	<meta>	John Miller		47	11/18/2025 09:10:50 PM
35	Admin	layout	Assets.fmp12	<meta>	John Miller		48	11/18/2025 09:10:50 PM
36	Admin	field	Assets.fmp12	<all>	John Miller	136	49	11/18/2025 09:10:50 PM

Whenever one of these catalogs is touched, you receive a JSON record with the following values:

account	The account name making the change.
baseableid	Optionally the table id if a table is modified.
catalog	One of the catalog names in the list above.
file	The database file name.
id	The ID of the item. May be for the list of items and for metadata of the item.
user	The user name.
counter	A value added by the plugin to give each record a unique number.
time	The time stamp when the plugin received the notification. The actual change may have been done a second earlier and the time you query the changes is probably some time later.

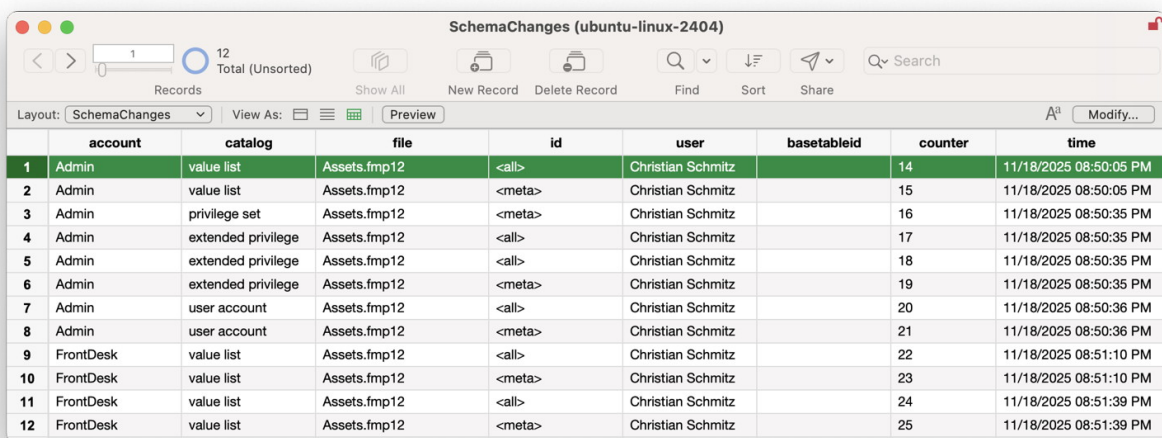
Start watching

On the server in the start script you call [SchemaChange.Enable](#) at least once. For example you can call it in the start script of the solution on the server. Once you enable the feature, our plugin starts collecting the JSON records for schema changes.

You can of course run this via PSoS on the server. The callback for the plugin only runs within the server script engine. While you could call the functions on the client, they would simply do nothing.

To disable, you can run [SchemaChange.Disable](#) in a script.

If you like to disable the possibility to disable it, you could use the [Plugin.LockFunction](#) function to put a lock on that function.



	account	catalog	file	id	user	basetableid	counter	time
1	Admin	value list	Assets.fmp12	<all>	Christian Schmitz		14	11/18/2025 08:50:05 PM
2	Admin	value list	Assets.fmp12	<meta>	Christian Schmitz		15	11/18/2025 08:50:05 PM
3	Admin	privilege set	Assets.fmp12	<meta>	Christian Schmitz		16	11/18/2025 08:50:35 PM
4	Admin	extended privilege	Assets.fmp12	<all>	Christian Schmitz		17	11/18/2025 08:50:35 PM
5	Admin	extended privilege	Assets.fmp12	<all>	Christian Schmitz		18	11/18/2025 08:50:35 PM
6	Admin	extended privilege	Assets.fmp12	<meta>	Christian Schmitz		19	11/18/2025 08:50:35 PM
7	Admin	user account	Assets.fmp12	<all>	Christian Schmitz		20	11/18/2025 08:50:36 PM
8	Admin	user account	Assets.fmp12	<meta>	Christian Schmitz		21	11/18/2025 08:50:36 PM
9	FrontDesk	value list	Assets.fmp12	<all>	Christian Schmitz		22	11/18/2025 08:51:10 PM
10	FrontDesk	value list	Assets.fmp12	<meta>	Christian Schmitz		23	11/18/2025 08:51:10 PM
11	FrontDesk	value list	Assets.fmp12	<all>	Christian Schmitz		24	11/18/2025 08:51:39 PM
12	FrontDesk	value list	Assets.fmp12	<meta>	Christian Schmitz		25	11/18/2025 08:51:39 PM

Scheduled script

To pick up the changes, we install a script and run it as a scheduled script. In the scheduled script we can query the SchemaChange.Changes function. We get a JSON arrays with all the collected change records. The scheduled script may run every minute or every hour and then process the JSON:

```
# check for pending changes
Set Variable [ $changes ; Value: MBS("SchemaChange.Changes") ]
#
# if we have changes, log them to a table
If [ Length ( $changes ) > 0 ]
    Set Variable [ $r ; Value: MBS("JSON.InsertRecords");
Get(FileName); "SchemaChanges"; $changes ) ]
    # table must have the following fields: account, catalog, file, id, user,
basetableid, counter and time
End If
```

In our example we have a table to store the records for the modifications. A table with account, catalog, file, id, user, basetableid, counter and time fields. The counter can be a number, but the others are text.

You could decide in the script what to do depending on what file or what account is making the change. Like you could send an email when some specific thing is changed or trigger the [Save a Copy as XML](#) script step to export the file.

Limits

The change records tell you what changed on a high level view point. Not much detail is given. Since you only get IDs for items like tables or scripts, you need to collect the information yourself to add a name to the IDs. Either by having a script run for each database file to query the schema and fill in records or by reading in a what is saved as XML from a database.

Our plugin with the [SchemaChange](#) functions can't know the details. Permissions prevent your script to query the database schema of other files unless you add them as data sources. But you can of course collect this in your own database for your own files.

Please try and let us know how well it works.