Key values

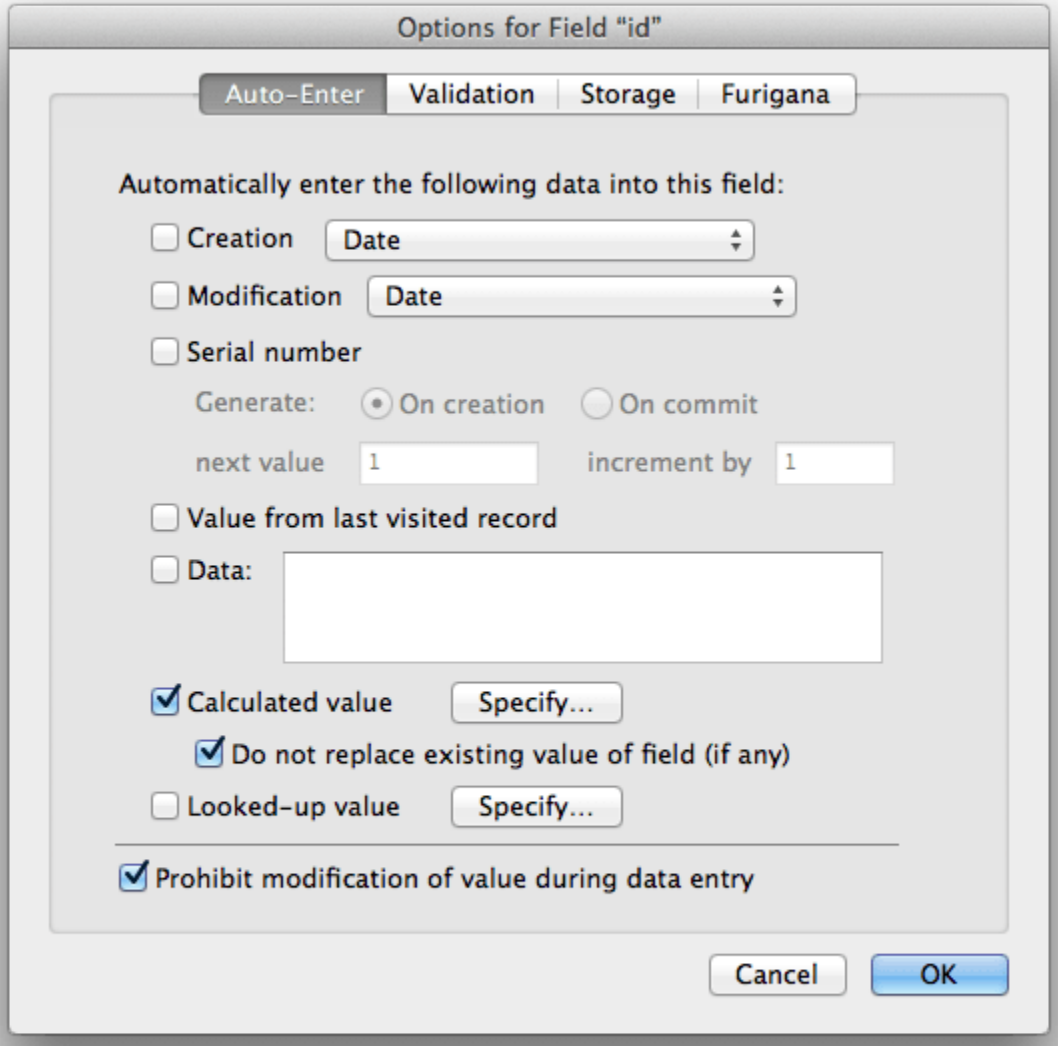
When developing in FileMaker, your default option for key values is FileMaker's auto-enter serial numbers. While this option can use an alphanumeric value instead of strict serial numbers, the general consensus for these documented standards is to use a universally unique identifier (UUID). Using a UUID scheme for creating keys nearly eliminates the chance of creating duplicate keys, which is a common problem when migrating data to new files (and remembering to reset auto-enter serial numbers) and when merging data created on multiple devices.

> ⚠ **Warning about duplicates**
>
> By default, the native FileMaker environment provides the **Duplicate Record** option within the **Records** menu. If one of the goals in using a UUID is to eliminate the possibility of duplicates, then a few extra steps will be required. See the best practice section below.

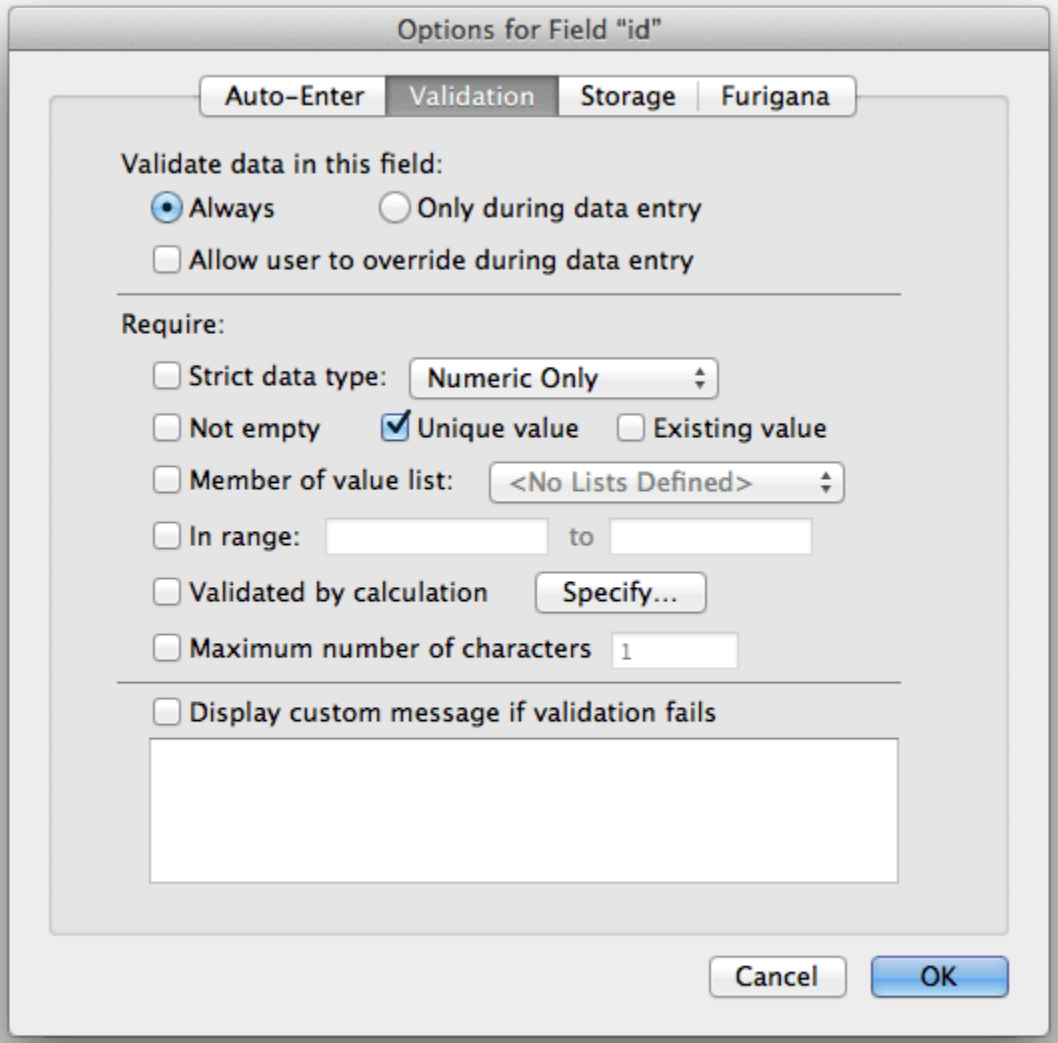
Best Practice

It is suggested that a wrapper custom function be used when integrating UUID values. A wrapper function is simply a Custom Function with a unique name which points to another chunk of code, internal or external function - see method stub. The suggested standard name for this reserved function is **UniqueID**.

In FileMaker, UUIDs are generated using one of three ways: FileMaker's Get (UUID) function, plug-ins, or custom functions. Any one of the methods below may be referenced within the **UniqueID** custom function. When using a UUID for your key value you may also want to follow the settings in the following dialogs.



In the above image the Calculated value is using the suggested **UniqueID** custom function. The option of "Do not replace..." is checked to prevent the replacement of the original value used. The "Prohibit modification..." option is set to prevent any accidental changes to the key value (should it be exposed to the end user - which is not suggested).



The validation options for the **id** key field have been set to require a "Unique value" and to **ALWAYS** validate by unchecking the "Allow user to override..." and changing the default setting of "Only during data entry". Leaving the "Not empty" option unchecked is optional as the auto-enter nature of the field implies that a value will be inserted (even when using ODBC). By setting these options, any action, either by the user or via scripted alteration, if a duplicate key is used, the file will prompt for record reversion. This type of setup compels the solution developer to ensure the uniqueness and canonical state of the key field. Features, such as the **OnObjectValidate** script trigger and using **File > Manage > Custom Menus...** (e.g. removing the Duplicate Record menu item) are methods one can employ to achieve the desired goal.

Supporting Record Duplication

By far, the most ensured way of avoiding any possible duplicate ID values is to use the settings above. Of course, FileMaker's validation will always catch any duplicates based on the "Unique value" validation. The method of supporting the creation of duplicate records, or cloning, is typically best handled via an OnRecordCommit layout trigger and through the use of a scripted process.

It is ALWAYS suggested that record cloning be managed via a script within the user interface. However, using a script trigger is inherently tied into the user interface and is layout or object specific. If a lower level method of duplication support is required then the following modifications may accomodate this.

In order to support FileMaker's native record duplication feature with UUIDs, just one change to the above dialog boxes must be made. The "Do not replace existing value of field (if any)" checkbox must be unchecked.

So long as no other fields are reference by this calculation, it should only generate a new UniqueID when a new record is created or duplicated. With the "Prohibit modification of value during data entry" checked, the user cannot modify the field value even if the ID field is shown and accessible (which, in most all cases, it should not be).

UUID Options

Get (UUID)

In version 12, FileMaker introduced the Get (UUID) function, which creates a unique 36-character string of mostly random hexadecimal digits according to the RFC 4122 standard. For example:

```
03E1BB14-B04C-4ACF-A1ED-C90E59EE0396
```

Plug-in based UUIDs

ScriptMaster by 360Works

The ScriptMaster plug-in by 360Works can call on Java standard libraries to generate UUIDs. The demo file included with ScriptMaster contains an example generating a random (version 4) RFC 4122 UUID, which looks like this:

```
4bf08957-6c8e-465a-946c-aa7b36d831cd
```

Custom Function based UUIDs

Custom functions for creating UUIDs are more portable than plug-in based UUIDs. Though you do need FileMaker Pro Advanced to install custom functions in a file, well-designed functions will execute correctly in all FileMaker platforms without the need to distribute and install a plug-in, to create a server-compatible version of a plug-in, or even the ability to run a plug-in in the case of FileMaker Go.

For security reasons Apple removed the ability for iOS apps, including FileMaker Go, to access the device NIC address in iOS 7. FileMaker Go running on iOS 7 will simply return "02:00:00:00:00:00" for the NIC address, regardless of the actual NIC address. Historically, some FileMaker UUID functions used Get (SystemNICAddress) as a component of generated UUID values, but this leads to a significant risk of different devices creating duplicate values at the same time, so any UUID functions using Get (SystemNICAddress) are now strongly discouraged. The Get (PersistentID) function introduced in FileMaker 12 is an acceptable substitute.

Tom Robinson's UUID (Type) function

Tom Robinson's UUID (Type) function creates hexadecimal UUIDs consistent with the international standard scheme. By passing the function different parameters, you can generate version 1 UUIDs (encoding timestamp and NIC address) or version 4 UUIDs (encoding random numbers). A sample (version 4) UUID looks like this:

```
f47ac10b-58cc-4372-a567-0e02b2c3d479
```

Calling this function with UUID (1) or UUID ("1h") will attempt to use the Get (SystemNICAddress) function, which will not return a correct result in iOS 7 or later, potentially leading to duplicate values. So using this function to generate type 1 UUIDs is strongly discouraged.

Jeremy Bante's UUID function

The UUIDTimeDevice function by Jeremy Bante (maintained in the fmpstandards Github repository) encodes the creation timestamp (UTC), part of the device's persistent ID, and a self-managed serial number. The generated values can be stored in a number field, which has certain performance benefits, and are meaningfully sortable by creation timestamp. Related functions are available for extracting the encoded timestamp, and for converting values to the conventional hexadecimal representation. A value from UUID looks like this:

```
120635243042352340000078541714672837368889
```

For situations where the privacy of a device persistent ID or a record creation timestamp is a concern, there is a related UUIDRandom function, which generates values like this:

```
42141279241146726753204187162615025729536
```