MDM Expression Engine: Validations, Assignments, and More...

Applies to:

SAP MDM SP4 Patch 3 HF 3

Summary

This article is a technical presentation of how best to mix and match, and utilize the various features of MDM and the expression engine in a central data management as well as consolidation scenario.

Author(s): Savi Sharma

Company: Nike Inc.

Created on: 27 March 2007

Author Bio

Savi Sharma has been with Nike for the past 6 years during which she played a role in all aspects of Data Management. She has equally good knowledge and expertise on the business processes, and information organization to match the business processes. Since 2005, Savi has been working on MDM in the role of MDM SME and Lead team member on the project implementing MDM as product information hub.

Savi has doctoral Degree in Biochemistry from INDIA, during which she has extensive data acquisition, analysis, and programming experience on experimental data involving proteins and enzymes.

Table of Contents

Introduction

I have been using the expression editor for implementing business rules for data validation for the past 2 years on Nike MDM implementation. The Data Manager Reference guide has only a brief explanation for each of the functions. When it comes to the actual nuts and bolts of writing expressions, however, you need a little more than that.

While implementing the business rules in a central data management scenario that deals with the main table, and with flat and qualified look up tables, I have discovered many rules, tricks, etc. on how best to mix and match, and utilize the various features of MDM and the expression engine. Here is a technical presentation of some of what I have learned for the benefit of other MDM project implementers.

Validation Expressions

Validation expressions provide a great way to enforce business rules and integrity constraints in central data management or consolidation scenarios. These are nice little snippets of code that can do wonders in many situations.

I have written expressions that enforce business rules starting with required fields (at the repository level as well as for a slice of master records), conditionally required fields (using 3 level deep nesting), length checks, date validations, status checks, etc. on look-up, main, and qualified tables.

- A discovery on real type fields that MDM “real” data type is a 4-byte real type field, and hence has an accuracy of 6 to 7 significant digits is an interesting one. I had a requirement to validate the length of 15 (digits) for a real type field, and had to work around this limitation.
- If you delete all validations from a group, and execute the group, you will get “There were no validation errors” message. Watch out!

- As of SP4, when you are writing validations on a nested look up table, in the "look up" tab, you will see all other look up tables in your repository, not just the ones that are in context of your look up table. So, if you want to write a constraint on a data field from a 3rd level nested table, you can use the look up values. I am hoping this is not a bug!

In my opinion, the expression engine has a lot of potential and uncovered features that can certainly play a role in taking MDM to the next level to become a data hub. Below, I summarize with examples some of the capabilities, and outline some of the limitations that I came across.

Expressions that can differentially execute on record Adds vs. Updates:

Imagine a business requirement where you need to validate the status of a look-up value on the main record for adds but not for updates. This has been something of a challenge until I came across the [Is New Checkout] function. The expression reads something like *If([Is New Checkout], Category.Status)*. This function works in concert with how iviews data maintenance is designed to work.

- Iviews check out record for data entry. (In case of Data Manager, you need to check out manually).
- If it is a record add, at "save" the function fires up.
- Remember, on a checked out record, even if you set automatic=Error for the expression, it issues only a 'warning' but that should be sufficient to alert the user to correct what they have entered.
- If the user fails to correct, iviews save also attempts to check in the record, and then it issues a hard error.

For validating updates, there are 2 beautiful functions [Is Normal Checkout], and Original Version. Again, imagine a business requirement where Gender of the main record cannot be updated once the product is activated. The expression reads something like *If ([Is Normal Checkout] AND IS_NOT_NULL(Gender) AND Active=True, [Original].Gender.[Record]=Gender.[Record]).*

This is much cleaner way of differentiating adds and updates than comparing create and change timestamps that I tried to use before the Checkout, and Original functions were introduced in expression editor.

Tips: 1.One important thing to remember with MDM is that create timestamp is initiated as soon as you click 'add'. Change timestamp is initiated at point of save, and if you have 40 fields to enter, they could be significantly different even for a first time add of a record.

2. Checkout does not work on look-up tables as of SP4. This may be a useful enhancement in a future release.

3. If you have expressions that use any of checkout/Original functions, and if you perform check /repair operation, check identifies them as "invalid token", and repair invalidates them ("invalid" added in front of these functions in the expression). This is communicated to be fixed in SP5 Patch 1.

Relationships and Expressions:

MDM relationship table is exposed in expression engine editor, and this enables business rule implementation and data validation for records in main and Qualified tables that are otherwise 2 distinct physical records.

- A caveat is that this requires the entry of the relationship itself into MDM relationships table.
- In a scenario where data is integrated into the MDM repository that is NOT a big deal.
- In a central data management scenario, however, it is a conscious decision to allow a business user to enter data into the relationships table.
- It is slightly more complicated than regular data entry. *As of SP4 Patch 3, MDM does not expose relationships table through iviews for data maintenance.*
- There is a great potential in exposing MDM relationships table for data entry through iviews.
    - Another interesting way would be making the relationship entry automated behind the screens by exposing a check box on the main record to indicate if a relationship should be enabled.

Sidebar: Exposing relationship data in Syndicator would be another useful thing.

- If main table and QLTs have parent/child data, then to syndicate out say product record along with some of its parent information is a 2-step process with the current syndicator (syndicate out parent and merge with child).
- This is not very conducive to automated event generation.
- On the other hand, relationship table data in syndicator would readily let you mix and match fields from parent and child, and from main and QLT to generate monolithic event that can serve multiple subscriptions.

Expressions involving Qualified Table (QLT), and Qualified links:

Using relationships, I have written a wide variety of expressions involving main and qualified table fields (non qualifiers and qualifiers). In that process, I realized the capabilities and limitations of the expression engine when it comes to qualified tables. I have always felt that while new features in MDM mostly work on main and simple look-up tables, their implementation on qualified table links and fields is almost an afterthought.

Some of the business requirements that are readily satisfied by using relationships in expression writing:

- A business requirement that validates the main table parent record field with the main table children records' fields for integrity
- A business requirement that defaults values on child(ren) records from the parent record for a subset of master records

If(div.record="x" AND recordind.[Record]="child" AND IS_NULL(field1.[Record]), ***parent to child.field1.[Record****], field1.[Record])*

- A business rule that compares qualifier or non-qualifier field values of child with parent record field values from the main table. In this type of comparisons, *HAS_ALL* is the function to use. Simple "=" does not work as we are dealing with qualified links from the main table context.
- A business rule to enforce the integrity between parent and child records' key fields in a qualified table. As long as it is only **one key field**, it is possible to compare between parent and child qualified links by simply using *HAS_ALL* function. If your requirement is to compare more than one key field (country and season fields between parent qualified links and child qualified links), MDM hits what appears to be an architectural limitation.
- Aggregate functions (count, sum, max, min, etc.) that need to roll up data from qualified table (qualifiers) to main table field work like a charm.

Current limitations of Expression Engine:

- Business requirements that **check for existence of a qualified table record** based on a pre-condition involving a main table field: IS_NULL/NOT NULL functions to check the existence of a qualified table record **throw erroneous results**. In fact, the expression fires regardless of whether the field in pre-condition is populated or not. Hopefully, this will be fixed in a near future release.
- MDM does not allow writing **assignments to qualifiers** (of a QLT) on the main table. You can write the assignment itself only on the qualified table, since the qualified table fields are NOT exposed from main table in the "Assignment field".
    - However, the assignments written on qualified table cannot be executed (and are not even exposed) from the main table record. And it is NOT meaningful to execute the assignment on just a non-qualifier value(s) because one non-qualifier can correspond to 100000 qualified table records. If you execute anyway, it gives a failure message.
- **Automatic validations involving qualifiers** are currently possible either via manual execution or using check out, and validate on check-in. Hopefully, this will be fixed in a near future release.
- Validation expressions that compare a main record field to a non-qualifier field of a qualified link **do not fire automatically** when main record and the QLT link are **added** at the same time. However once added, if you execute the expression manually or if you make an update, the expression fires up.
- Expressions that need to operate **at individual qualified link level are limited by MDM architecture**. MDM considers all the qualified links as multiple values of the qualified look-up field on the main table. So, a comparison to ensure that a parent has the same country and season combination relationship in the QLT before a child record is added in the QLT does not work.

  - A HAS_ALL function always returns a positive result as long as the country and season you are looking for exist individually on 2 separate qualified links (criss-cross match between links). To the best of my knowledge, MDM does not have plans to address this in the near future.
- **Aggregate functions cannot operate across 2 qualified tables** since again MDM considers the 2 qualified tables as 2 multi-valued fields on the main record. The functions also cannot operate at individual qualified link level. To the best of my knowledge, MDM does not have plans to address this in the near future. An example is a requirement to roll up effectivity dates from an item-country-season QLT to an item-country QLT.

Systime() Function

I think Systime () function will in the future be very handy to compare today's date with a date field's value. A business requirement that says if an effective date is less than today's date, fields b and c need to be populated (*If (systime(0) –eff date >= 1, IS_NOT_NULL(b))*) or a requirement that an effective date must always be greater than or equal to today's date should be possible using systime function.

- In the current MDM release (SP4 Patch 3), the same function gives spurious results with ">=" operator. It works fine with simple greater than (>) operator.
- It also gives false negative results if you compare the result to '0' (*effective – systime(0) > 0*) or if you equate systime(0) function to any value.
- You also cannot write an expressions that equates systime(0) to a field. For instance, if ">=" does not work, technically it can be written as *effective dt=systime(0) OR effective dt –systime(0) > 1*
- Net result of above limitations is that you can only validate a date that is greater than today's date by at least 2 days.
- Since **Literal Date fields** do not have a time component, they **are treated as 12AM** or 00:00:00.000 Hence, the expression *effective dt - SYSTIME(0) > '-1'* should work. This will use **GMT** (UTC) time coordinate.

  - You can change the factor in the parenthesis according to your time zone. For example, you can use *SYSTIME('-0.3333333333')* for PST.

In my opinion, limitation around comparing with ">=" operator should be addressed in a future release.

Systime(0) works great with assignments. If I want to default system date to a field on the main record but only for a slice of total set of master records, systime(0) does it. I can default based on another associated field value

If(IS_NULL(field1) AND IS_NOT_NULL(field2), systime(0), field1)

or based on the same field being null- *If(IS_NULL(field1), systime(0), field1)*

For a business requirement that needs the state transition date defaulted to system date at item add as well as every time an item undergoes a state change, you can use the [Is New Checkout] and [Is Normal Checkout] in conjunction with systime() to implement the rule. You **want to** default the date **only if the update is on the state** field during record updates.

If([Is New Checkout]AND

IS_NULL(date) AND IS_NOT_NULL(state.[Record]), systime(0),

IS_NOT_NULL(date) AND [Is Normal Checkout] AND

***[Original].State.[Record]<> State.[Record]**,*

Systime(0), date)

In my opinion, systime() function has a lot of additional potential for solving a variety of business requirements around validation and assignments involving date fields.

Using Expressions in Search on Data Manager and Syndicator:

If you want to use an expression in your search for a slice of main and qualified table data either for viewing or syndication, you need to follow the search priority order: first filter using your expression, and then use the filters (look-up fields or Booleans). If you do it the other way round, (as soon as the expression executes) the previous filters are wiped out.

Assignments:

Assignments are one of the most wonderful inclusions into MDM. They are very useful in a central data management scenario as well as in consolidation scenarios.

- Master data repositories often have records from multiple business units, and records that are parent-child (in the absence of 2 main tables).
- The fields on main records as well as QLT records have values that may differ by business unit. For one business unit the field value is a calculation or derivation involving multiple fields; for another line of business it could be a value coming from another system (via integration); for a third business unit it could simply be a default value.
- Calculated fields lock the field to be a calculated ONLY and not updateable by any means (integration or data entry).
- Assignment expressions let you default, derive, calculate, and integrate values into the same field differentially by any parameter (business unit, parent/child, etc.).

Assignment expressions let you default only when the field in question is NULL. If the user decides to enter a value, then they let the user entered value be. To default the value, the valid value list for every look-up field is exposed.

If(Div.[Record]="x" AND IS_NULL(Category.[Record]), Category[NOT APP], Category.[Record])

A key characteristic of assignment expressions is that the else clause must always be specified or else it will give erroneous results ✍

- Lookup Fields should be assigned lookup values (using the Lookup menu from your expression editor) and cannot be assigned a specific String value. In contrast, text fields cannot be assigned a lookup value.
- If you need to run an assignment on a multi-lingual field, the use of the language check from the expression engine lets you specify the language in which the value should be assigned: "If(Language = English[US],...)

One last but very important note: assignments do not fire automatically on record adds or updates. They need to be manually executed. The way to automate assignment execution is by using workflow.

Assignments and Workflow:

Assignment workflows are very simple to design. Each workflow has 3 steps, Start, Assign, and Stop.

- Assign step is the place to attach the assignment expression that you want to execute in the workflow.
- You can assign only individual assignment expressions, and not a group. This is unlike validation expressions where you can assign a validation group to the "validate" step.

One current limitation is that one workflow needs to be designed per assignment. In other words, if you string together multiple assignments in a single workflow, it results in a "SQL command execution error" on the database. One efficient way to do it is to attach the next work flow at the 'launch' property of the stop command of previous workflow.

- The trigger action of a workflow needs to be set to "manual" to be visible in the available workflows tab of the launch property of the stop step.

The assignment calculation is performed automatically via the launch of the workflow at each record update, including of a field not participating in the assignment expression. So, every assignment expression must include a condition on triggering the workflow and the assignment.

- The common and simple conditions to include are: calculating the value of the assigned field only if it is blank, or basing the assignment on other fields' values or the NOT NULL condition.

As of SP4 Patch 3, automatic assignment execution using workflow gives an "information out of date" error on record add. Record update results in workflow getting stuck in error.

- In case of cascading workflows, in record update scenario, if you manually send the workflow stuck in error to next step, then first workflow of the cascade does not result in any assignment, even though it shows completed status.
- But the remaining workflows in the cascade finish successfully, and values are assigned to fields.
- Once you add cascading workflows, in a record add scenario, right click assigns all values accurately!

Assignments Workflows & Override Values: If there is a business case for user override of assigned value (calculated via expression+ workflow) for a given field, you will need to add an additional field called "override flag" to enforce the conditionality on when the manual override is possible. Here are some more details:

- If an assignment field's value is manually overridden, then the workflow triggered will be stuck in 'Error' on the Assignment Step.
- If the record is updated with new values in other fields than the ones participating in the assignment calculation, the overridden value prevails, and a new workflow is triggered and becomes stuck in 'Error' on the assignment Step.
- If the record is updated with new values in the fields participating in the assignment calculation, then the assignment field value is re-assigned automatically: a new workflow is triggered and completed successfully.
- If in the business scenario the assignment value is calculated first, then the override value is never re-calculated, you can use the following trick in your assignment calculation: *"IF(IS_NULL(Assignment Field.Record),.assignmentexpression ,Assignment Field.Record)"*

Assignments and Security

Below is a summary of observations on MDM security- role and privilege set up for assignment expressions.

- When a field is updated via an assignment embedded in a workflow, the user id field's value is [System]. Change history, however, will give the accurate representation of change by user id.
- If you are in a Read-Only mode (no execution allowed and read-only on all tables and fields) and attempt to execute an assignment, then the message returned is "Assignment Operation Failed" and not "Insufficient Rights for the Operation". Apparently, this is normal for MDM, so watch out!
- If you are in a Read-Only mode then you are still able to add a record manually to a workflow, and hence to execute the assignment. Existing workaround is to not use the trigger action "Manual" when you design your workflow. Apparently, this is normal for MDM, so watch out!
    - The assignment step that is activated from a workflow is not user specific, it is owned by the system. If you add a user stamp you can see that. What is missing is the ability to block the "Add to Job" function to a read only user.
- There is no way to allow or prevent specific assignment operations from the Security Tab. The assignments operations are currently part of the validations table. In my opinion, a security model that restricts end-users from executing the validations and assignments manually works best in such cases.

Assignments Involving Multiple Field Logic and Concatenation Function

The assignment engine is powerful enough to execute recursive logic involving multiple main record fields, in deeply nested if, then, else loops.

For concatenation, my business requirement was to string 4 fields together with a specified delimiter, and assign the result to field Z. This Z field was length constrained to 20 characters. Therefore I needed to check the length of the final string, and if it was more than 20, I needed to specify the logic and priority order for involving the fields in the concatenation.

- Concatenation function *concat()* strings fields together with a default delimiter of ";".
- MDM does not permit configuration of the delimiter to another value (say, "-" or a "/").
- I also realized that concat() cannot be used as the value to be assigned into a field in an assignment expression.
- Functions like *LEN()* on top of *concat()* function do not work.

I found the solution in the *ampersand* ("&") function that can string take the delimiter of my choice as another string value, and can do the job. The final string can be length checked and assigned.

```
If (Div=xd AND IS_NOT_NULL(field1.[Record]) AND IS_NOT_NULL(field_2.[Record]) AND
LEN(field1.Desc& "/" &field2.Desc& "/" &field3.Desc& "/" field4.desc) <=20,
(field1.Desc& "/" &field2.Desc& "/" &field3.Desc& "/" field4.desc),
If (Div=xd AND IS_NOT_NULL(field1.[Record]) AND IS_NOT_NULL(field_2.[Record]) AND
LEN(field1.Desc& "/" &field2.Desc& "/" &field4.desc) <=20,
…
…
)))
```

Tips: 1. Compiling expressions that have multiple if, then, else loops nested works ONLY if you successively add the nested if, then, else statements. If you write all of it at once, the compiler (save operation) fails, and gives you an error message "Syntax error". In reality, it is NOT due to a syntax mess-up; it could possibly be because the engine fails to compile recursive loops. Since you cannot save the expression engine code until it is saved, you lose the code when you "Esc" out of the window.

2. Once you save a long assignment, you will find, when reopening, that code formatting is completely messed up. Some code is put one long line, and others will be left alone. You will also find that the spacing between operators and fields is deleted. Reformat it if you can. On such occasions, I also noticed that some AND operators were blacked out (color of the font). This results in syntax error. Just delete the AND operator and reselect it.

3. When you use Boolean fields in assignment expressions, you need to equate them to TRUE or FALSE, even if you assigned your TRUE value to be Y and False value to be N in your console. Otherwise, the assignment executes but falls into the else clause and assigns a spurious value.

4. Another very interesting finding is if you first test for 'all negative condition' of a given assignment, it gives "Assignment operation failed" message. However, if you test a positive condition first, and then the negative, the same condition that gave a failure message gives a success! My experience is with something like this:

```
If (IS_NULL(field A) AND IS_NULL(field B), "NA",
If(IS_NOT_NULL(field A) AND LEN(field A& "/" & field B) <=8,
(field A& "/" & field B), field A))
```

After saving this, I tested with both fields A and B null, and looked for NA to be assigned into my field in question, and it gave me the failure message. I tested the second condition, and then came to the first one, and it worked fine.

Expression Engine of My dreams:

Addition of several small features would make the expression editor much more user friendly.

- Currently there is no ability to save partially written code or code that is throwing syntax errors (false errors, many times). With addition of assignments as a new feature, you can write very complex logic involving multiple fields. Ability to save would let the MDM technologist not to lose time writing and rewriting the code.
- Ability to blink at the piece of code that is in error would be a huge bonus. Currently, the editor reformats the code, and sometimes the operators (e.g. AND) are disabled. And it starts throwing syntax errors which does not get us anywhere.
- Ability to compile recursive loops would be a boon. Currently, if you plan to write a complex assignment logic involving multiple main table and qualified table fields, you need to write each if , then, else loop, compile, then write the else piece, then compile, and so on. This, and the fact that field names, and field values (in case look-up fields) need to be picked from the tab eliminates the option of writing code in a note pad.
- Ability to configure properties for some of the functions of the expression engine.
- Many expressions that work well with main table fields fail to work in case of qualified table fields (qualifiers, non-qualifiers or the qualified links). It would be nice to see the expression engine on the same scale of maturity for both main and qualified tables.
- Last but not least is the ability of the expression engine to respond to metamodel changes. Currently, if you change a field name, field type, delete a field, etc. there is no simple way to figure out which expressions are impacted unless you change the expression as soon as you are done changing the model (in the console). The expression engine reacts/responds very inconsistently to metamodel changes. When you execute the expression that has field(s) that underwent metamodel changes, sometimes the error message is 'no validation error', and at other times it is syntax error. I have also seen MDM respond by preventing any record add/update and throwing out a blanket “validation/calculation failure happened”.

Aspiring a little higher, assignments that can search and return a value into a look-up field, taking the values of 2 other look-up fields as input or ability to use variables, and reuse the values would be great!

Acknowledgements:

Caroline Debattista and I shared the joy of prototyping expressions with SP3.2 software, and then preliminary work on assignments and workflow on SP4 patch2. Many thanks to Caroline for our work and discoveries together.

This article is written using results of testing conducted on SP4 Patch 3. *I might submit a follow-up article if I see significant enhancements for some of the tool’s limitations covered in this article.*

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.