One of the most exciting features of AwareIM is it's execution Context.
Once you'ver understood how this magic "Execution Context" is formedÂ most data processing is a snap.
I assume that you've read and (mostly) understood the section "How Context is Formed" from the AwareIM documentation.

For example you can run a process that says

```
FIND BOExample WHERE BOExample.Attribute1='Yes'
UPDATE BOExample
```

This is a simple process that causes rules attached to BOExample to execute.
For example such a rule could contain:

```
BOExample.Attribut2=CURRENT_TIMESTAMP
```

What's happening isÂ a SQL statement like

```
UPDATE BOExample SET Attribute2=CURRENT_DATE WHERE Attribute1='Yes'
```

OK, 3 lines of AwareIM code compared to 1 SQL statement. But let's take a look on how we could manage this in MS-Access code. (I just typed the code here – so I don't know if the syntax is 100% correct)

```
set rs=currentdb.openrecordset("select * from BOExample where Attribute1='Yes')
do until rs.eof
rs.edit
rs!Attribute2=current_timestamp
rs.update
rs.movenext
loop
```

You might ask what Access code has to do with AwareIM – well I'd like to explain AwareIM's execution context based on a quite common and easy understandable language…

FIND BOExample …. does what set rs = currentdb… / do until rs.eof does. It finds all desired records and puts all of them into the context. Afterwards it executes UPDATE BOExample for every BOExample in context. Just like my do/loop construction above. Calling UPDATE BOExample triggers the execution of the rules attached to BOExample. This rule now has it's own execution context with only 1 instance in it.

Now let's assume that we don't want a simple assignment of CURRENT_TIMESTAMP to Attribute1. Now we want to find data from some other Business Object. Our rule could read like

```
FIND BOTest WHERE BOTest.Attribute1 = 'abc'
BOExample.Attribute2=BOTest.Attribute2
```

When the rule gets executed then there's the 1 instance of BOExample in it's context. Now we find a BOTest and this BOTest also gets put into the context. Afterwards it gets assigned to BOExample

One drawback of this approach ist that the rule doesn't only get executed when the process gets called. It also gets executed every time BOExample is updated. Either by user interaction or by some other Process that modifies BOExample.

Well, so I change the process to:

```
FIND BOExample WHERE BOExample.Attribute1='Yes'
FIND BOTest WHERE BOTest.Attribute1 = 'abc'
BOExample.Attribute2=BOTest.Attribute2
```

If you think that we're now done then you are wrong ! The process above will modify only the first BOExample ! Why ? Read on….

FIND BOExample places several instances into the context. After that AwareIM starts to call FIND BOTest for every BOExample, but FIND BOTest destroys the context created from FIND BOExample !
In case you are a little bit familiar with coding in Access then take a look at the following code:

```
set rs=currentdb.openrecordset("select * from BOExample where Attribute1='Yes')
do until rs.eof
set rs=currentdb.openrecordset("select * from BOTest where Attribute1='abc')
do until rs.eof
rs.edit
rs!Attribute2=rs!Attribute2
rs.update
rs.movenext
loop
loop
```

This code won't work too ! If you take a close look then you'll note that I've assigned the variable rs twice. The second "select * from …." gets assigned to the same variable and destroys the previous content. In Access it would be easy to solve. I'd just change it to

```
set rs=currentdb.openrecordset("select * from BOExample where Attribute1='Yes')
do until rs.eof
set rs2=currentdb.openrecordset("select * from BOTest where Attribute1='abc')
do until rs2.eof
rs.edit
rs!Attribute2=rs2!Attribute2
rs.update
```

```
rs.movenext
loop
loop
```

I only changed the second set rs=… to set rs2=…
In AwareIM it's a little bit more tricky.

Process1

```
FIND BOExample WHERE BOExample.Attribute1='Yes'
Process2
```

Process2 with BOExample defined as process input

```
FIND BOTest WHERE BOTest.Attribute1 = 'abc'
BOExample.Attribute2=BOTest.Attribute2
```

Now I have 2 processes. The first finds the desired BOExamples and puts them into the context. For every instance in context it executes Process2 and passes the current instance of BOExample over to Process2. That's why we need BOExample as input for Process2. Since there's now only one instance of BOExample in the context of Process2 finding instances of BOTest doesn't destroy the other instances in context.

I hope that I could shed some light on how AwareIM forms its context. Please feel free to register yourself on this site and post comments.