

QUERY-DRIVEN AUTOMATION USING GOLDBOX

QUERY-DRIVEN vs. EVENT-DRIVEN – As you probably know, **GoldMine's Automated Processes (APs)** are **Event-Driven**. In this paper, I'll be comparing GoldMine's Event-Driven automation to GoldBox's **Query-Driven Automation**. The differences are major, and almost entirely in favor of GoldBox's Query-Driven Automation. What's more, the more demanding and complex the requirements for automation become, the more the scale tips in favor of the GoldBox approach.

What is Contact Automation? However it's accomplished, Contact Automation involves checking specific **data conditions**, or "triggers", to evaluate whether the current data justify taking certain **actions** automatically, for each Contact checked. These actions may range from adding a Contact to a Group, to sending the Contact a certain kind of E-mail, to Scheduling a Call for the Contact...there are many, many more possible actions.

The difference between Event-Driven and Query-Driven Automation does not lie so much in the actions that need to be considered; they are about the same. The real difference lies in **how the conditions are evaluated**.

How GoldMine's APs work

When an AP Track is attached to GoldMine Contact John Doe (either manually, or by another Track), GoldMine evaluates (either immediately, or at the time of the next AP Scan) all the Preemptive Events of the Process (if there are any), and takes any actions called for by them. GoldMine then attaches and evaluates the first Sequential Event for that Process, and continues the processing through the Sequential Events in the Track until an Event is attached that **cannot be processed** (meaning that its **data trigger condition** is not satisfied). At this point, processing is suspended for Contact John Doe. If an AP Scan is in progress, the AP processing continues with the next Contact.

The next time APs are scanned, GoldMine repeats the process for Contact John Doe, starting with evaluating the Preemptive Events for the AP(s) then attached. Processing of the Sequential Events is generally linear, except when an Event (often, but not necessarily, a Preemptive Event) calls for "switching tracks". Let's look at the process of switching Tracks more closely.

Generally, a single AP Event has, at most, two trigger conditions: the "main" **trigger**, and the optional **trigger filter**. In situations where **many conditions** should all be satisfied before an action ought to be taken, APs are **entirely dependent on the Track itself** to apply those conditions, **cumulatively**. In a sense, an AP Track is a **progressive series of filters**, with Actions carried out at appropriate points. It is often assumed that the triggers of all the Events that preceded whatever Event is currently being evaluated were all satisfied, and **are still satisfied**.

Naturally, that last assumption is not always reliable, so there has to be a way to deal with the possibility that a Contact previously satisfied the trigger for some previous Event, but now does not satisfy that filter. GoldMine deals with this possibility via Preemptive Events.

For example, suppose a Contact acquired the **Marketing Process** at the time the Contact's **Type** (Key1) changed to **Hot Prospect**. As long as the Contact is proceeding through the Sequential Events of the Marketing Process, you might think there would be no need to keep verifying that the Contact is a Hot Prospect. But what happens when the Contact actually buys something, and so changes Type to **Customer**?

We could use a Trigger Filter on every Event, to check that the Type value remains Hot Prospect. But better, we can use a Preemptive Event that checks the Contact Type. However, what if a Sale was made, but somehow the Contact Type value never got changed? We can use a different, more powerful Preemptive Event to check for the existence of a certain type of History record (a Sale). That's something a Trigger Filter cannot do. In fact, you can have the Preemptive Event branch to a special Sequential Event, if a Sale record is found; and have that special Sequential Event change the Contact Type, and proceed to another Event that will switch the Contact to another AP Track, the one for Sales.

If it seems that reading the last half dozen paragraphs has gotten just a bit tedious for you, well, that's what it's like dealing with APs. While the concepts of APs aren't really so difficult, working with them is demanding. The order of the Events that make up the Tracks; and the components of each Event; these must be "just so". Documenting the APs while writing and testing them can be a real challenge.

But above all, dealing with "train wrecks", however they occur, can be very problematic with APs. Determining which Contacts have been affected by a specific problem; and then determining exactly how to correct the problem; these can be tough to do.

How GoldBox Query-Driven Automation works

With QDA, the first job is to list the Actions that need to be automated. This is essentially the same set of outcomes that we'd deal with if we were using GoldMine's APs, except that many tests (like for the existence of an E-mail Address) that are Actions with APs are simply part of the query with QDA. Also, while QDA does not strictly require that we name our "processes", like GoldMine APs do, it's a good idea to do so, anyway. For each process, we list all the Actions we may want to take.

For each Action, we create a GoldBox setup that will accomplish that action. For example, any time we want to Schedule something; or add a History record or a Linked Document; we'll want to use a **GoldBox Import** of one kind or another. If we want to edit a database field, we'd use a **Global Replace** or an **Update**. GoldBox has a setup that will accomplish everything we might need in the way of an action, with one exception: sending an E-mail. But even with that, we can use GoldBox to attach (Import) a GoldMine AP Event that will send that E-mail.

Once we have the "what to do" lined up, we need to determine "whom to do it to". Here is where the simplicity of QDA comes to the fore. **For each action setup, we write a query to retrieve the correct list of Contacts to whom that action will be performed.** Then we "feed" those Contacts to the GoldBox setup.

Here's a fairly heavy-duty example: Suppose we have an E-mail that we want to send only to Customers who have purchased Product A and Product B but not Product C. This is for a new division, so we only want to send it to Customers in 5 specific States, or to people who have one of ten specific Area Codes. For our initial broadcast, we want to send only to Customers who have a specific Type of Detail record attached to their Contact record; or who have one of 30 specific job Titles. (In future broadcasts, we'll remove the Detail record restriction and increase the list of Titles to 50, but we need to make sure that we don't send the E-mail to the same person twice).

If someone came to me and asked to set up a GoldMine AP with those conditions, my answer would be simply **"No"**. I doubt that it's even possible, but I **know** I don't want to touch it. But with QDA, it's **"No problem"**. Queries are powerful things.

So, with each action, we use a query (usually in the form of a Convert to dBase operation that will create the Source file for the action's Import Setup). Now all we need is a way to automate the running of all these queries and setups...probably at night...probably every night. We just string them all together into one or more GoldBox Q-file scripts; then have Windows Task Scheduler run the (first) Q-file whenever we want. **Now, we have every Contact in the database being evaluated for every possible action, every night.** Such comprehensive testing sounds like overkill; but in fact, it's just a natural result of the way QDA works.

Query-Driven Automation is inherently self-documenting, as a process. But to help things along, I've created a set of GoldBox Views to help build the system collaboratively. You can read about that on my site, at <http://www.goldboxbob.com/examples9.html>.

And, QDA can readily be used to embellish Contact records with detailed information about the status of a Contact record within each "process". But above all, it's the power of the queries, the extreme flexibility that they provide, that make QDA a clear winner.