

INTERNATIONAL  
**SPECTRUM**

THE MULTIVALEUE TECHNOLOGY MAGAZINE | SEPTEMBER/OCTOBER 2019



# Creating The Next Generation

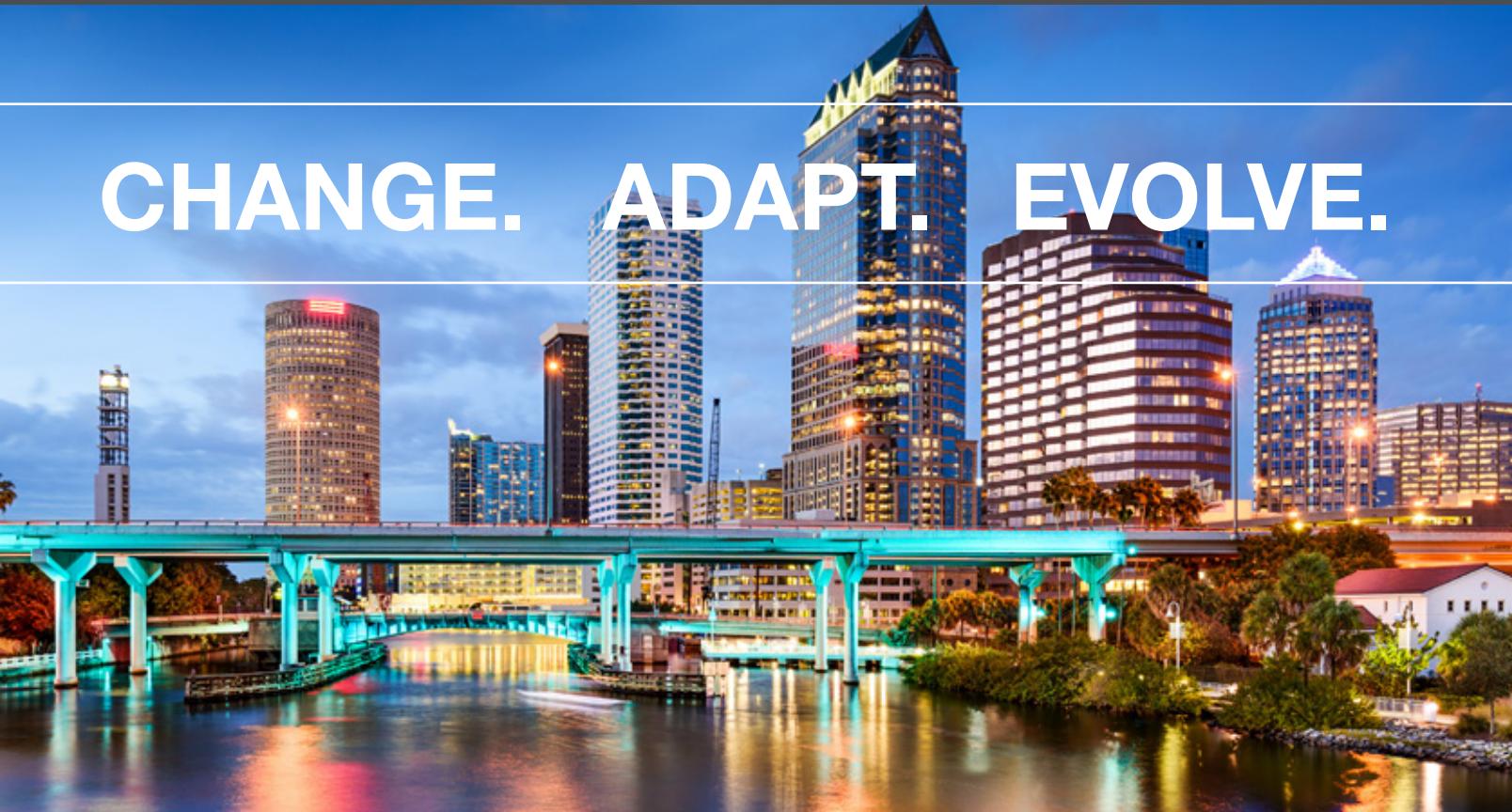
**ALSO IN THIS ISSUE:**

- The NACHA Cha-Cha
- MV Reporting: A New Tool
- Leveraging MV to Learn Everything



INTERNATIONAL  
**SPECTRUM**  
MultiValue Conference and Partner Exchange

**CHANGE. ADAPT. EVOLVE.**



**39<sup>TH</sup> Annual Conference**

APRIL 20 – 23, 2020 | SADDLEBROOK RESORT, TAMPA, FL

— **SAVE THE DATE** —

EARLY BIRD REGISTRATION STARTS

**OCTOBER 28<sup>TH</sup>**

[www.intl-spectrum.com/conference](http://www.intl-spectrum.com/conference)



INTERNATIONAL SPECTRUM



SEPTEMBER/OCTOBER 2019

## COVER

6

**Business Tech: Building MultiValue Programmers**

- **Part I** We hear it all the time: I can't find MV programmers. As someone who has trained a lot of programmers over the years, I assure you that there are qualified people out there. And, if you need more, it isn't that hard to create them. One of the reasons MultiValue survives is because people with a technical or business background can learn it quickly. A good, solid set of lessons can go a long way. Here's the first one. **BY CHARLES BAROUCH**

## FEATURES

10

**Making NACHAs: Part I** Electronic Funds Transfer is a broad term for a critical process designed to move money in a controlled, well-documented manner. Core to this mission is the NACHA file format. Like EDI, this format was created as a simple, precise, and exact method. Like EDI, everyone has elected to implement it a little differently. This series will help you understand how to dance the NACHA cha-cha. **BY KEVIN KING**

14

**Basic MV Reports – A New Way to Handle U2 Reporting** As the "islands of information" model of business continues to implode, we are more and more tasked with developing methods for sending and receiving data. Sometimes, the modern answers can be found with old friends. Farley Welch makes the case for using the dBASE data model as a twenty-first century transport for your MultiValue data. **BY FARLEY WELCH**

20

**The Rosetta Stone Project** The OCONV is an amazing conversion tool. Imagine if you could pack it up and take it with you outside the MultiValue world? Aaron Young, Dick Thiot, and I decided to do just that. In this series, we'll show you code in several other languages that does parts of what OCONV does. Use it to teach yourself other languages or use it to bring the best of MV to all your projects. **BY CHARLES BAROUCH, WITH ADDITIONAL CODE BY AARON YOUNG AND DICK THIOT**

## DEPARTMENTS

**From the Inside** page 4**From the Press Room** page 12

International Spectrum and MultiValue are registered trademarks of International Spectrum, Inc. All other registered trademarks are the property of the respective trademark holders.

# From the Inside

**A**rtificial Intelligence is a hot topic for businesses. As with most big things, it hasn't come to the forefront because of rigorous research. It has come because of the success of other projects like it. It's already transforming several industries. It has been all over the news.

Despite what the growing excitement implies, AI is not a new concept. This hot new thing has been with us for quite a while.

There's a Spectrum Magazine article, written back in the early '90s, that shows you how to create an "Expert System," a form of AI decision making, inside the PICK Database. To be fair, the introduction of much more powerful computers and specialize chips has also driven the uptick in attention. AI software tends to be very calculation heavy. Faster processing, disk, and RAM have all extended the reach of AI.

While most CEOs or CIOs are aware that it exists, many are still unsure how to employ it to their benefit. Knowing AI makes robotics, drones, image recognition, and driverless cars more viable is only useful if you are in those specific markets. When innovation comes from executives seeing it on the news, it tends to lack clear links to the business they are operating.

Additionally, a lot of AI software is outside the reach of most companies' resources and budgets. Without clear plans

and serious benefit analysis, attempts are likely to be money pits, not successes.

It comes down to a fundamental confusion over exactly what AI software is and what it can do.

I like to explain it by dividing AI onto three separate components: Data Analytics, Predictive Analysis, and Machine Learning. And yes, I'm lumping high-speed data mining in for simplicity's sake.

True AI is all about Machine Learning. But this is the most sophisticated form of AI and likely not going to be used in business software. Those edge cases are growing and becoming less edge-like, but they aren't center-stage today. Data Analytics and Predictive Analytics are really what businesses should be focusing on.

Data Analytics is all about finding trends in your data. Most enterprise software has so much data. No one know where to start to get answers without someone asking smart, focused questions. While this sounds a lot like Data Mining, the difference is that Data mining returns the specific results, already formatted for processes like dashboards and reports. While Data Analytics is designed to take raw data, and look for any patterns outside of the already understood Dashboards. One pushes data into assumed relationships while the other derives relationships from the data.



NATHAN RECTOR  
*President*

CHARLES BAROUCH  
*Editor*

TRACEY RECTOR  
*Layout*



*Learn more about the MultiValue Symbol and see what MultiValue Technologies and MultiValue Communities exist to help you support and manage your business and systems. To find out more visit*

<http://www.intl-spectrum.com>

---

**MISSION STATEMENT** International Spectrum magazine's editorial mission is to be the premier independent source of useful information for users, developers, and resellers of MultiValue database management systems, open systems business database solutions, and related hardware, software, and peripherals. Published bimonthly, International Spectrum provides comprehensive coverage of the products, companies, and trends that shape the MultiValue marketplace as well as the computer industry at large — helping its readers get the most out of their business computer systems.

---

International Spectrum is published six (6) times per year at the subscription price of \$40.00 U.S. in the U.S.A.; \$45.00 U.S. in Canada and Mexico; \$50.00 U.S. for other countries. Single copy rates are \$7.00 U.S. in the U.S.A. and Canada, and \$9.00 U.S. in all other countries. International Spectrum is published by International Spectrum, Inc., 3691 E. 102nd Ct., Thornton, CO 80229; Tel: 720/259-1356; Fax: 603/250-0664 E-Mail: [request@intl-spectrum.com](mailto:request@intl-spectrum.com). Copyright 2019 International Spectrum, Inc. All rights reserved. Reproduction in whole or in part, without written permission, is prohibited.

PRINTED IN USA

**NEWS RELEASES/UNSOLICITED ARTICLES**  
International Spectrum is eager to print your submissions of up-to-the-minute news and feature stories complementary to the MultiValue marketplace. Black and white or color photographs and diagrams are welcome. Although there is no guarantee a submitted article will be published, every article will be considered. Please send your press releases, articles, and queries to: [editor@intl-spectrum.com](mailto:editor@intl-spectrum.com). International Spectrum retains all reprint rights.

International Spectrum is a registered trademark and MultiValue is a trademark of International Spectrum, Inc. All other registered trademarks and trademarks are the property of the respective trademark holders.



Data Analytics is also used to support or debunk existing assumptions.

Predictive Analytics, on the other hand, is all about taking the data you have and projecting into the future. If you have an inventory control and purchasing system, then you are likely already using an older form of Inventory Forecasting. These forecasting reports are using a Predictive Analytics model.

The main difference between these existing reports and the newer Predicative Analytics AI processes is the complexity. Modern Predicative Analytics take into account far more data complexity. They address What If questions, rather than just estimate future needs based on past performance.

AI Machine Learning is the Holy Grail; and much more complex. It combines the concepts of Data Analytics and Predictive Analytics,

but instead of the person asking a specific question, the computer returns its own assumptions based on the data provided.

This is why Artificial Intelligence and Machine Learning are so important for business. You provide your data to an AI, and instead of have a person who knows how to ask a precise question, you have the computer providing these assumptions for review. As we all know, people are better at picking apart an answer than at providing a complete one.

Make sure you join us at the International Spectrum 2020 Conference in Florida to talk more about how Artificial Intelligence will be used with your MultiValue database.



**NATHAN RECTOR**

President

International Spectrum

nathan@intl-spectrum.com

## International

## Spectrum

### 2018 Compilation

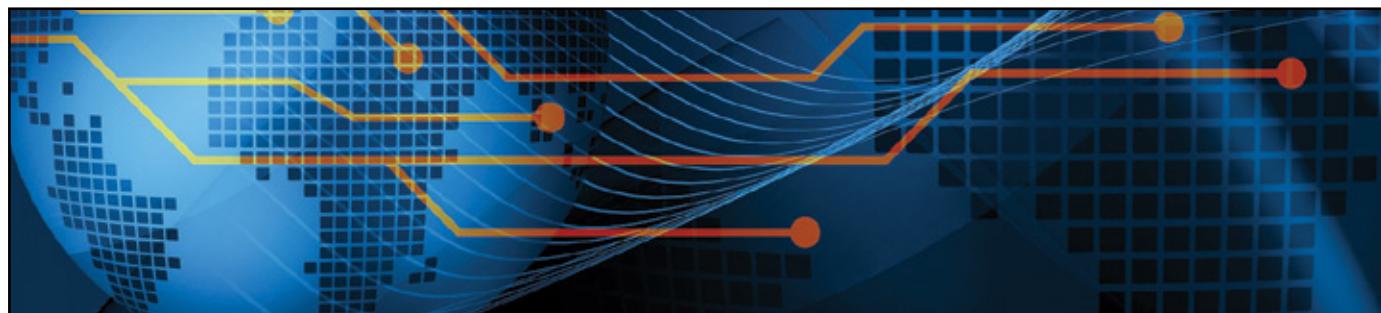
**Now Available in Print!**

January/February 2018 to November/December 2018



In a Single Bound and Printed Volume

Available on Blurb  
Search for  
International Spectrum  
in Bookstore



## GET CONNECTED

**KNOWLEDGE AND EDUCATION FOR THE MULTIVALE PROFESSIONAL**

Professional Memberships provide you access to knowledge, solutions, information, and code that you won't find in other locations.

**Membership Includes:**

- Magazine
- Newsletter
- On-Demand Videos
- Discounted Conference Rates
- Research papers
- Case Studies
- Source Code



<http://www.intl-spectrum.com/membership/>

# Building MultiValue Programmers

## Part I

BY CHARLES BAROUCH

**N**ot everyone is a natural-born teacher. This article series is not a substitute for real training. If you have the knack, this series is here to help you build your own programmers. If you have no choice — can't engage a trainer — at least you'll have this.

When I teach, I like to start with the basics. Even if the person I'm teaching is experienced in other systems, laying groundwork helps them connect with my perspective. Sharing a common view makes teaching easier.

### What is Data

When computers were first entering small to mid-sized businesses, data was an obscure concept. Today, while many people lack a solid, practical definition, the idea itself is already firmly

---

### *SQL's rules are wildly insufficient.*

---

established. Training a new MultiValue programmer or analyst is both easier and harder because of this.

Let's start by mixing how data is represented with what data is. While explicitly separating those ideas makes academic sense, merging them makes it graspable. You'll see very little academic sense in this series. We aren't building a theology, we are building for practical use.

#### Fixed

Fixed length data is a bedrock concept. Look at this example [Figure 1].

1234567890123456789012345678901234
ROBERT    MAY           POLITICIAN    58BLKBLK099887765

First Name	ROBERT
Last Name	MAY
Profession	POLITICIAN
Age	58
Eye Color	BLK
Hair Color	BLK
ID #	099887765

Figure 1

The problem with fixed length becomes obvious when Alexandra Rogers has to be added to the database. We've only allotted seven characters to the First Name field. She needs nine. To fix this we have to re-factor the entire database to allow nine characters in the First Name and change every reference in every program.

When she divorces and goes back to her maiden name of Wychowski, we have to do the same thing for our too-small Last Name field. And, if we get an eleven character long job designation, again we re-factor.

To make matters worse, Robert's full name is now taking up eighteen spaces because every name must take up the same amount of space. Yes, disk is cheap, but everything you do to make data less efficient should come with a meaningful benefit. This does not.

#### Delimited

Delimited fields solve the length problem [Figure 2].

Oops. We lost a field (Age) on Alexandra. In a fixed format, that would have stood out visually. In delimited,

we don't have neat columns. We need to insert her age, 26, to fix the data. The good news is that our data only takes the space it needs, plus one for each delimiter. We did have to trade a some human-readability.

### **SQL**

SQL tries to solve the missing field issue by adding validation rules directly into the database schema. In this case, Age would have been an Integer field which would have stopped us from adding BLU as the value.

Unfortunately, SQL also reintroduces maximum field lengths, which is a major step backwards. This might be a good trade, except that SQL's rules are wildly insufficient. Yes, knowing that Age is numeric would have caught the error on our previous example, but generally, knowing if something is a number, boolean, or character, is an extremely limited validation. You still have to do real validation in the programming layer, so SQL's half-thought-out feature is just a way to split the logic arbitrarily into two places.

### **Our Story so Far**

All three of these models are predicated on a poor premise: One question equals one answer. What if Robert works a second job? What if Alexandra has a second ID #? In the real world, data is rarely one to one. These are flat data models. Real data is lumpy.

ROBERT, MAY, POLITICIAN, 58, BLK, BLK, 099887765
ALEXANDRA, WYCHOWSKI, MATHEMATICIAN, BLU, RED, 986753099

First Name	ROBERT
Last Name	MAY
Profession	POLITICIAN
Age	58
Eye Color	BLK
Hair Color	BLK
ID #	099887765

**Celebrating 20 Years  
as a leader in the  
MultiValue Industry**



D3 UniVerse UniData PICK jBase mvBase Caché

### **IT Solutions. Proven Results.**

**One Project. One Decision. One Keystroke at a time.**

- *Custom developed solutions based on your needs*
- *Senior level developers and business analysts to guide you*
- *Developing long term partnerships*

**www.pickprogram.com  
contact@pickprogram.com  
(614) 921-9840**

If I'm making dolls, I might have a retail price, a wholesale price, a samples price (free), and a donation price for tax purposes when I send my excess inventory to a charity. In fixed, delimited, or SQL models, I have a choice of making each of those a discreet location in my data, or I have to resort to child tables [Figure 3].

With this extended structure, we now requires five reads to get a product and price. Yes, CPU is cheap, but everything you do to make data less efficient

should come with a meaningful benefit. This does not.

Of course, if you have another multiple-answer situation, you spawn another child table and add some more reads. Perhaps the doll comes with several possible outfits. This is worse than the price issue because you not only have additional reads but you also have a variable number of reads, creating an

SQL Product Table
Product ID #
Betsy 9988772
JoeJoe 9988773

SQL Price Table
ID # Type Price
9988772 W 10.00
9988772 R 15.00
9988772 S 0.00
9988772 D 12.00
9988773 W 9.50
9988773 R 13.25
9988773 S 0.00
9988773 D 11.75

**Figure 2**

“are we done yet?” inefficiency where you have to scan until you don’t find any. In a child table with a large number of rows, this matters.

## XML/JSON

XML or JSON might be possible solutions [Figure 4].

With XML and JSON, we’ve lost the easy scanning of fixed fields. We’ve also lost the relative compactness of delimited rows because we are now required to add the field tags into every single record. This is an example of a terrible trade-off. It is less human readable, it is less computer readable, it is space ex-

```
<?xml version="1.0" encoding="UTF-8"?>
<dolls>
  <product>
    <id>9988772</id>
    <price>
      <type>W</type>
      <amount>10.00</amount>
    </price>
    <price>
      <type>R</type>
      <amount>15.00</amount>
    </price>
  </product>
  <product>
    <id>9988773</id>
    <price>
      <type>W</type>
      <amount>9.50</amount>
    </price>
    <price>
      <type>R</type>
      <amount>13.25</amount>
    </price>
  </product>
</dolls>

JSON
{
  "dolls": {
    "product": [
      {
        "id": "9988772",
        "price": [
          {"w": "10.00", "r": "15.00"}
        ]
      },
      {
        "id": "9988773",
        "price": [
          {"w": "9.50", "r": "13.25"}
        ]
      }
    ]
  }
}
```

Figure 4

pensive, and slower than every format previously discussed.

## Columnar

Columnar databases (like Hadoop) have a smart premise: We search more than we write, so let’s optimize for searching [Figure 5].

By splitting the data by column, we are back to multiple reads but unlike SQL, we get something in exchange. When I search by profession, I only deal with one table, and that table has the smallest amount of data needed to resolve that part of the query.

Additionally, in a ten field database, while I do have to do ten reads, I get the ID from the first read and that changes the reads from searches (fixed child tables, delimited child tables, SQL child tables) to targeted reads (columnar tables). And yes, SQL has indexes, but so do Columnar databases. So instead of the index being a speed-up for the inefficiency in SQL, in Columnar our indexes are a speed-up to an already efficient system.

## Document Databases

Document databases (like MultiValue and MongoDB) are based on a different premise than the ones above. They support embedded table logic to allow the flexibility of child tables without the extra reads.

Now, I need to say this before we go any further. Document databases can do what every data format we’ve listed above can do. We can implement columnar logic, fixed logic, delimited logic, or SQL logic in a document database. We can do this easily. Further, we can mix approaches.

Negative: This is chaotic! Positive: This is flexible and models the real world.

For this example, I’m going to use MultiValue because MongoDB represents as JSON, which we have already covered. Additionally, I will use MultiValue’s preferred method because all of the other methods it can do are covered above [Figure 6].

First Name Table	
1	ROBERT
2	ALEXANDRA

Last Name Table	
1	MAY
2	WYCHOWSKI

Profession Table:	
1	POLITICIAN
2	MATHEMATICIAN

Figure 5



# MERCURY FLASH



**run your database  
efficiently**

**the ULTIMATE**  
file management tool for Rocket  
Software's UniData and UniVerse  
databases.

[www.paradigm-systems.us](http://www.paradigm-systems.us)  
**561-705-3688**  
[sales@paradigm-systems.us](mailto:sales@paradigm-systems.us)

ROBERT^MAY^POLITICIAN] BRICKLAYER^58^BLK^BLK^099887765  
ALEXANDRA^WYCHOWSKI] ROGERS^26^BLU^RED^986753099]123557722

**Figure 6**

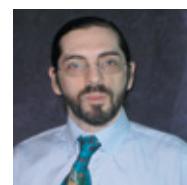
Robert's additional job (Bricklayer) is accommodated by adding a multi-value to attribute three. That's the terminology we use. Alexandra's two legal last names share an attribute. Her two ID #'s share their attribute.

So, scorecard: we have the advantages and disadvantages of delimited text, but by embedding delimiters within delimited text, we have acquired single reads. Mike Ruane calls this compressed XML. The attribute numbers stand in for the XML or JSON tags, creating compactness while also keeping space efficiency.

### **What do I use?**

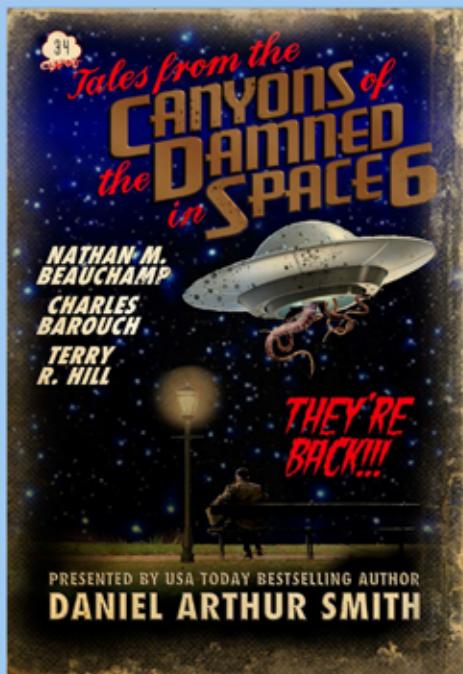
I use everything. I prefer MultiValue in most cases, but the goal of a database is

to store, manage, and return data. That can be done with all of these. If a shop is already using Hadoop, use it. If they have SQL, use that. If they have Multi-Value, use that. **IS**



**CHARLES BAROUCH**  
is the CTO of HDWP,  
Inc. and the Publisher  
at HDWPbooks. You  
can read his writing in

International Spectrum, Theme-Thology,  
Novo Pulp, Pax Solaria, PerehelionSF, and  
the Interrogative series, which begins with  
Tiago and the Masterless.



**On Kindle**

**Paperback, Nook, Kobo, and  
Audiobook coming soon.**

### **Amazon Top 10 Hot New Release**

This thirty-fourth issue of Tales from the Canyons of the Damned consists of four sharp, suspenseful, thought provoking short stories—each from a different featured master of speculative fiction.

**Exclusionary Symbiosis** by Nathan M. Beauchamp

**Ship of the Dead** by Charles Barouch

**Last Visit to the Park** by Terry R. Hill

**Off-World Kick Murder Squad VII** by Daniel Arthur Smith

Tales from the Canyons of the Damned ([canyonsofthedamned.com](http://canyonsofthedamned.com)) is a dark science fiction, horror, & slipstream magazine we've been working on since 2015. What is Dark Science Fiction and Horror? Think of it as a literary Twilight Zone, Night Gallery, or Outer Limits, it's Netflix's Black Mirror and Amazon's Electric Dreams in the short story format. And it's a bargain. Each monthly issue has three-to-five sharp, suspenseful, satirical tales from today's top speculative fiction writers.

These are Dark Sci Fi Slipstream Tales like you've never read before.

# Making NACHAs

## Part I

In 2018, the ACH (Automated Clearing House) network processed nearly 23 billion payments, according to [www.nacha.org](http://www.nacha.org). Note this isn't payment AMOUNTS, but rather payment COUNTS. The amounts of these transactions, in a single year, exceed \$50 trillion. Ponder that for a moment... \$50 trillion of money moved in one year — over a million per second!

What started as the backbone of the U.S. banking industry, the ACH network — also commonly referred to as Electronic Funds Transfer (EFT) — is now integrated into most industries, providing a service to move money from one entity to another securely. Whether it's receiving or sending money, direct deposit of payroll, or just about anything involving the movement of money, you can bet the ACH network is probably in play somewhere.

One of the most common ways to communicate with the ACH network

*...even though it's an established standard, each receiver implements things just a little bit differently.*

is through a text file known as a NACHA (National Automated Clearing House Association) file. The format is ridiculously simple, in theory. But, because it supports all sorts of different types of electronic transactions, building a program to generate a NACHA file can be an especially onerous task. That's the point of our discussion today; let's try to demystify some of the weird NACHA-isms.

The first thing to understand about the NACHA file format is that even though it's an established standard, each receiver implements things just a little bit differently. You'll need to work with your trading partner (the party sending or receiving the file) to make sure what is sent is properly interpreted on the other end.

In this series, we're going to build a file that is used to send money to people while at the same time receiving money from people. For this reason, it can be useful to think of the NACHA file like a checkbook; you're going to write checks to give money to people, and you're going to have deposits of money coming into your account(s). The NACHA file standardizes this into a transmittable format.

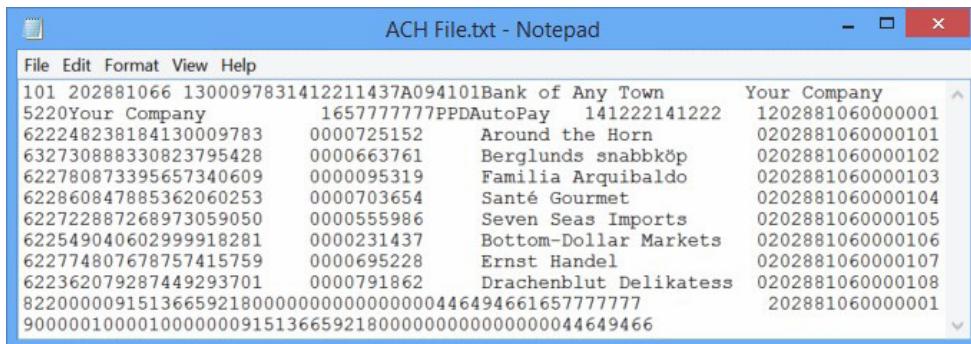
Let's take a peek at a sample NACHA file[Figure 1].

That's clear, right? I'm sure that if you've never seen a NACHA file before, this is just a wall of text. Let's break it down, brick by brick.

The key to understanding any of it is to know that each line is 94 characters. A line is not necessarily a complete record as we think of records.

The first character of each line is a line type. Here's a translation guide:

- 1: File header
  - 5: Batch header
  - 6: Detail
  - 7: Addenda (not shown)
  - 8: Batch footer
  - 9: File footer



**Figure 1**

# IT audits have you jumping through hoops?



**PRC** can help you meet your compliance requirements and make IT more agile and productive. No extra work, nothing to remember, nothing to fall through the cracks. Our software development lifecycle tool automatically prevents or detects change according to your criteria. You can deploy, rollback, test and report quickly, automatically and with confidence. Let PRC protect your company's valuable U2 data and software assets.



SJ+ Systems Associates • info@sjplus.com • http://sjplus.com



stuffed into each 94-byte record. To better understand what's inside of them, we first need to review a few NACHA terms. From my own experience, having done several NACHA extracts, the terminology is typically one of the biggest tripping points.

If you look up the terms *originator* and *receiver* on the various websites which explain NACHA, the vague descriptions will not be helpful. Instead, returning to our analogy, think of the *originator* as the person with the checkbook. If you're sending the file, you are the *originator*. Or more accurately, your bank is.

However, if you're receiving the file, you are the *receiver*. Keep this in mind, because it has nothing to do with whether you're the one sending or the one receiving money, it is about who is sending and who is receiving the file.

If you build the file, you're the *originator*. The accepted synonyms are *source* and *company identifier*. Starting to see how this can get confusing?

That's why I wrote this. I've already stubbed my toe on these issues. I want to save you from the same pain.

The term **ODFI (Originating Depository Financial Institution)** is the source of much of the confusion. If

you're sending money, your bank is the ODFI. That just makes sense. However, if you're receiving money from someone else, the sender's bank is the *originator*, right? WRONG. If you're building the file, the ODFI is the ABA (American Bankers Association) number of your bank. Looking again at our analogy, the ACH network needs to know where our bank account lives. The ODFI provides that value.

The ABA is the 9 digit number on the bottom left of a check — surrounded by funny graphics — is commonly known as a *routing number* or *transit number* or *routing/transit number*. In ACH parlance, however, it's commonly known as an ABA number. It's all the same, but like most things in ACH, there's a bunch of different terms to describe it.

Following ODFI is RDFI (**Receiving Depository Financial Institution**). Just think of RDFI as the other guy. This can also be known as the *destination*.

The final term to become familiar with is the *Center Name*. This describes the ACH endpoint who will receive your file — commonly a bank. Some banks don't care about this value, whereas others are very, very specific about the center name and will reject the file if

the center name is not exactly what they want.

Before we get into the details of building each record, there is one more important detail to cover. Once you've built your NACHA formatted file, then what? You'll need some way to transmit this to your assigned center. On Multivalue systems, this is commonly done using SFTP. If you are dealing with a bank which doesn't support SFTP, you'll need to download the file to a workstation and then upload it to a web form provided by the bank. Regardless of how the information gets out the door, it is imperative to always keep in mind that this is the sensitive information and must be protected in every way possible. This is **your** banking information and the banking information of your customers and vendors, and there are countless nefarious people on the web sniffing the wire for exactly this kind of information to do as much damage as possible.

Believe it or not, I have seen banks that want the NACHA file emailed to them. Don't do it. It would be more secure to copy the file to a flash drive and drive it to the bank. Then again, a flash drive isn't exactly secure either.

In later installments of this series, we'll break down each of the different types of NACHA file records and explain more confusing terminology and how you can easily generate this type of file on any Multivalue system. **IS**



**KEVIN KING** is the President and Chief Technologist with Precision Solutions, Inc., a leader in technology solutions, support, and training.

# FROM THE PRESS ROOM



## MVExtensions 2.0 Released for Visual Studio Code

The MVExtensions Team has published MVExtension 2.0 for Visual Studio Code. MVExtensions is a Visual Studio Code used for syntax highlighting, IntelliSense and program formatting for Pick-Basic code development.

Download and install this Community Project from:

<https://github.com/mvextensions/mvbasic> ■



## Paradigm Systems Releases Mercury Flash V5.2

Paradigm Systems will release version 5.2 of their Multivalue database management system Mercury Flash in early October.

Starting with this release, OpenQM 3.4-16 by Zumasy will now be supported in addition to the UniData and UniVerse databases from Rocket Software.

"We are excited to add OpenQM to the list of supported databases for Mercury Flash. OpenQM is the first addition to our supported environment and we are in the process of creating versions to support other Multivalue environments, said Jay LaBonte, president and founder of Paradigm Systems.

In addition to supporting OpenQM 3.4-16 in above in this release, there are some additional improvements users will find in Mercury Flash 5.2 are:

- Networked sensors such as Temperature, Humidity and Barometric pressure can now be monitored.
- The User Defined Scheduler has been greatly improved allowing additional security on commands that can be executed within each defined schedule.
- A new Export button has been added to several utilities allowing the report contents to be exported to a PDF formatted report and then printed.
- Account Caching has been improved to better handle distributed files.
- Improved handling of virtual accounts.
- Improved processing by the Account Cleaner to better identify items for cleanup.
- A new Honey Pot utility has been added to allow you to monitor unused ports and identify scanning bots and other malicious software

that is accessing your network.

- Various minor fixes and improvements.

Mercury Flash is the state-of-the-art web based management console specifically designed for the UniVerse, UniData and OpenQM databases. Mercury Flash version 5 was released in July of 2018 and over the past year it has experienced incredible growth and acceptance in the community and has quickly become the go to solution for Multivalue database management and tuning. ■



## Revelation Software Release OpenInsight 10.0.7

### Major Changes in the OpenInsight 10.0.7 Release

OpenInsight now provides an option to change the precision of mathematical operations. This can be set by calling the setEPMMode stored procedure, passing in 1 to enable and 0 to disable the functionality. By default, the extended precision math will maintain 32 digits of precision, but this can be modified by calling the setEPMMode-Precision stored procedure, passing in the number of digits of precision desired. Both the enabling of extend-

ed precision, and the default number of digits, can also be defined in the application properties. The following operators/functions are affected: +, +=, -, -=, \*, /, ==, =, !=, ◊, >, <, >=, <=, mod(), int(), abs(), atan(), cos(), exp(), ln(), pwr(), sin(), sqrt(), tan()

OpenInsight's RLIST functionality and performance have also been improved starting with the 10.0.7 release. A new version of RLIST (RLISTX), which optionally replaces and extends RLIST, is available to enable these enhancements. RLISTX merges the features of RLIST, SELECT\_INTO, OLIST/RUN\_REPORT, and RTI\_XBAND. One obvious change is the ability to pass in multiple select statements to RLIST in a single call, @FM delimited. Using the Record Editor, or the Configuration Record option from the OI Console, you must create a CFG\_RTI\_RLIST record in SYSENV, with RLISTX in field 1.

If this record doesn't exist, or has anything other than RLISTX in field 1, then normal RLIST behavior ("RLIST 9") and functionality will remain. (Note that the CFG\_RTI\_RLIST information is cached by your system; after changing this value, you should exit and re-enter OpenInsight).

The MultiValue BFS (MVBFS) connections for QM, D3 and U2 have been enhanced to submit multiple select lists to the "back end" host for bulk processing whenever possible. This enhancement can result in significant performance improvements when

# FROM THE PRESS ROOM

using an MVBFS connection. Note that these changes work in conjunction with the RLISTX changes discussed above; you must enable RLISTX and install a “plugin” stored procedure on the host system to access these changes. There is now a button on the MVBFS connection designer which will install this plugin (a program named RTI\_MVBFS\_SERVER\_PLUGIN\_U2, RTI\_MVBFS\_SERVER\_PLUGIN\_D3, or RTI\_MVBFS\_SERVER\_PLUGIN\_QM).

Starting with OpenInsight 10.0.4, “child” processes launched from OpenInsight can be configured so that they do not consume additional license seats. In particular, CTO, AREV64, BRW, and O4W calls (using the engine server’s built-in web server) made from a copy of OpenInsight will not count against the licensed count of users.

For example, a single user copy of OpenInsight can now run the IDE, a CTO session, and generate a BRW report at the same time.

Note that this enhancement requires both OpenInsight 10.0.4 (or above), and the Universal Driver 5.2 (or above). OpenInsight 10.0.7 will work with the Universal Driver 5.1, but it will not exhibit these license enhancements until it is “paired” with a UD 5.2.

To take advantage of this license enhancement for O4W and engine server tasks, users/developers must update their eserver.cfg file (either

directly or through the Settings dialogs). In particular, any passwords that are currently explicitly embedded in the eserver.cfg can be replaced with an asterisk (“\*”); this indicates that the specific connection should use the enhanced licensing. This also has the additional advantage of no longer requiring manual updates to the eserver.cfg file when passwords are changed for the applications or users defined in the connection string. (As an additional enhancement, the username can also be replaced with “\*\*” if you wish to use the ‘default user’ created for an application). ■



## Zumasys Release AccuTerm 8

The wait is finally over! AccuTerm 8 is here.

AccuTerm 8 brings new security, a fresh user interface, and the ability to run AccuTerm securely through your web browser.

AccuTerm8 features dozens of new features that give you access to your PICK system when and where you need it. Our new subscription model bundles Desktop, Web, and Mobile editions; plus, you’ll receive all future updates automatically so you stay up to date with the greatest features.

- New User Interface – Tear-off tabs, drag and drop windows and new Visual Studio Code color schemes.
- Enhanced Security – OpenSSL, brand new encryption libraries and the latest cryptography.
- Access from Any Device – Run your PICK application over the Web or in the Cloud with our new fully-responsive HTML5 browser interface.
- Plus new 2:1 licensing allows you to run

AccuTerm Desktop on two machines, like office and home; a new centralized administration provides control over user access; and enhanced session resilience (ReZume) restores dropped sessions running over the Internet, Cloud, etc.

The best terminal emulator for PICK just took a major leap forward but this is just the beginning. The new subscription model means that you can continue to obtain AccuTerm for a very low up-front price and receive continual updates and upgrades. Costs and budgets are controlled with a subscription and you know what you are paying every year. AccuTerm Web with subscription puts you on a path to the Cloud and Software as a Service which is the future.

Already an AccuTerm user looking to upgrade? Contact us today about some special upgrade discounts which are available until 12/31/19. ■

## e-Xtra Newsletter

### Stay on top of Industry News

- ◆ Tech Tips
- ◆ Job Postings
- ◆ New Products
- ◆ Corporate Updates



[www.intl-spectrum.com/newsletter](http://www.intl-spectrum.com/newsletter)

# Basic MV Reports

## A New Way to Handle U2 Reporting

BY FARLEY WELCH

There are many ways to report and analyze data contained in your U2 databases. I would like to share with you a simple process that overcomes many of the obstacles normally associated with analyzing MultiValue data. To meet my real-world needs, I have developed a callable UniBasic subroutine that will convert your U2 data to transportable DBF files (dBASE 4) that can be natively used by Excel as an ODBC source for pivot tables or reports.

dBASE is a venerable file format that supports up to a 2gb file size and can contain hundreds of thousands of records with up to 255 fields. These are theoretical values, but I've created practical applications with very large data sets without encountering any capacity issues.

Unlike CSV, dBASE is a binary format, so fields can be designated as text or numeric in a way which is recognized by Excel. dBASE files can be compressed. It is a very effective format for use with the string data stored in U2 databases.

**Basic MV Reports** (BMVR) helps you present Unidata data sets as Excel pivot tables. Pivot table templates can be distributed separately to your user com-

*It operates as a form of middleware dependent on your selected and formatted data.*

munity and conveniently reused when source data is updated on demand or by scheduled phantoms. The process can be integrated into any menu system that can call UniBasic subroutines.

The subroutine supports two dBASE file creation modes: FULL mode creates a stand-alone file that can be opened by Excel. INFO mode creates component elements of the .DBF format which can be combined with the stored results of other runs of the same report. This supports a form of data warehousing and can be very efficient when managing historical data.

You control the selection and formatting of your data. **Basic MV Reports** is not a report writer. It operates as a form of middleware dependent on your selected and formatted data. Multi-user support is provided through arguments that utilize the indexing you provide on your source data.

The steps to implement a report can be summarized as:

- Compile and catalog the BMVR.DBASE.ENGINE subroutine in any U2 directory
- Create Q-Pointers for all data files that are required for your report
- Run your UniBasic program to select, update and format your report data
- Call BMVR.DBASE.ENGINE with required arguments
- Handle the delivery of the resulting .dbf file to a local or network share
- Open your Excel pivot table template designed for a specific ODBC data source
- Refresh your Excel template

### Free is a Very Good Price

No strings, no fine print, I'll send you all the code and instructions needed to use this no-nonsense solution to your Unidata reporting requirements. There is no charge or obligation and you are free to use, modify and integrate these tools as you see fit. All code is in text format and contains no binary or compiled elements. Free really is a very good price, and the value is demonstrable.

Just email your request to: [basicMVreports@gmail.com](mailto:basicMVreports@gmail.com) and I'll get a package to you immediately.

My mission is to empower you to get more value from your Unidata/Universe databases without the high cost and complexity often associated with MultiValue reporting solutions. Utilizing Excel pivot tables as a presentation layer allows your users to expand their analysis without tying up your programming staff. One back-end solution can support many end-user requests.

If you're pleased with the value and have other needs, you're invited to reach out to talk about it. For more information, please check-out my website at: [www.basicMVreports.com](http://www.basicMVreports.com).

Thanks in advance for your consideration of Basic MV Reports. I look forward to hearing from you.

### **Source File Configuration and Multi-user Implementation**

Create a standard U2 file to hold the data for your report. The file must be sized to hold the maximum number of records you expect to include multiplied by the number of simultaneous users you expect. This is a temporary file that is cleared after the .dbf output is created. Each record is keyed using a unique session ID and a counter. Use @LOGNAME : @TTY concatenated with a numeric record counter to create a unique record key. Create an attri-

bute (typically, UNIQUE\_ID) on the file and INDEX this attribute. Store @LOGNAME : @TTY in this attribute.

Create dictionary attributes for each data element you intend to include in your report. Define each attribute with the position, length and data type as you normally would. All attributes are single valued. The display name (dictionary attribute #4) is used as the .dbf file column header. These names must be 10 or less characters long, must be in all CAPS and must not contain any special characters except the underscore. You can also create a new dictionary attribute for the .dbf column header and specify that attribute when BMVR.DBASE.ENGINE is called. This can be useful if you have other uses for your reporting file that require a more descriptive display name.

### **Populate the Reporting File**

One of the first and last steps in your data population routine will be to remove records from the report file for this unique user session [Figure 1].

Your report file population program can be simple or complex. You are in control of all joins, lookups, calculations and other data transformations that you require. When each reporting record is completed it is written to the report file using a key equal to @LOGNAME:@TTY:"\*":<counter>.

You might be asking yourself why the subroutine does not support virtual attributes since the ability to use I-type and V-type attributes is an important feature of U2. The short answer is "simplicity and performance". Using a dedicated and indexed source file populated using UniBasic is simply the most flexible way to manage large amounts of data. Your report population subroutine can calculate virtual attributes in line or duplicate the results using UniBasic as needed.

Yes, this may seem like a lot of additional work, but the results can be very well received by your users. Pivot tables provide users with the tools to get new insight from data sets and a single pivot table can satisfy numerous reporting requests.

Control the operation of the subroutine through the arguments in Figure 2.

INF – the name of the Unidata file containing the data to include in the output. This is a file constructed using your own UniBasic routine that accepts input; performs selects; normalizes multi-value data sets; and performs conversions and translations.

INFIELDS – dictionary items to include in the output. If set to "ALL" then all D-type attributes are included. Only D-type attributes are supported.

```
UNIQUE = @LOGNAME:@TTY
CLEAR.CMD = "SELECT <your report file> WITH UNIQUE_ID LIKE `:QUOTE(UNIQUE)`
PERFORM CLEAR.CMD CAPTURING SPONGE
CDONE = 0
LOOP UNTIL CDONE
  READNEXT CLEAR.ID ELSE CDONE = 1
    DELETE <file variable opened to your report file>,CLEAR.ID
  REPEAT
```

**Figure 1**

```
SUBROUTINE BMVR.DBASE.ENGINE(INF, INFIELDS, XKEY, XVAL, OUTF, OUTMD, RESULT)
```

**Figure 2**

## PICK/U2 Resources Available

Execu-Sys, Ltd is an Executive Search & Consulting firm that has specialized in the PICK/MULTIVALEUE market since 1988 and is the Preferred Partner of Rocket Software for PICK/U2 professional services.

Hourly rates for contract programming are extremely competitive and there is no minimum time or \$ commitment.

Contact us today to discuss potential engagements.

**Matt Hart**  
**EXECU-SYS, LTD**

1411 Broadway, Suite 1220  
New York, NY 10018

**(800) 423-1964 x302**

Email: [mh@eslny.com](mailto:mh@eslny.com)

XKEY – the name of an indexed field on the data file. This is a value unique to the current report and can reference a specific user session or a phantom calling the report. If the value provided

is set to the literal “SELECT-LIST” then the value in XVAL (next argument) is the name of the select list to use.

XVAL – value to use when selecting records from the source data file. If this value is multi-valued (delimited with CHAR(253)) then it is handled as a list of keys and no further select is performed. If the value in XKEY is “SELECT-LIST” then the value in XVAL is the name of the prebuilt select list which is retrieved. Otherwise the value is used as the indexed key. The type of selection is determined by the dictionary of the indexed field.

OUTF – path and name for the completed dBASE file. Value 1 is the name of the file; value 2 is the operating system path. Value 3 sets the operating system (W = Windows (default) or U = Unix). This setting determines operating system file copy and delete commands.

OUTMD – multi-value list of mode type switches that control the operation of the report generator.

If OUTMD<1,1> equals the literal “FULL” then the output is a full formatted dBASE file written with the .dbf extension. If value 1 equals “INFO” then the output consists of dBASE component parts that can be later combined and assembled into a complete file.

OUTMD<1,2> is set to “YES” if conversion formatting is to be applied to the output data. The default is “NO” which uses raw data from the source file.

OUTMD<1,3> is the dictionary field number to use as the column heading in the output. The default is attribute 4 which is the normal field name. Field names must be 10 characters or less, must be in all CAPS and cannot include some special characters. Using this switch allows you to specify a name for each column in the output without interfering with other uses of your source file.

RESULT – returns “OK” if all went as planned. This return argument is set to “ERROR” followed by an error message if a problem was encountered.

Code fragment to call **Basic MV Reports** subroutine.

Insert the code in Figure 3 into your report population routine to call the dBASE generation. Upon return, the .dbf file will be located per your specification.

The dBASE routine can handle a very large number of records without overflowing shared memory because the .dbf file is written to disk sequentially[Figure 4].

```
INF = "<your report source file name>"  
INFLDS = "ALL"  
XKEY = "UNIQUE_ID"  
UNIQUE = @LOGNAME:@TTY  
XVAL = UNIQUE  
OUTF = ""  
OUTF<1,1> = "<name for .dbf file>"  
OUTF<1,2> = "<path where you want the .dbf written>"  
OUTF<1,3> = "U"  
OUTMD = ""  
OUTMD<1,1> = "FULL"  
OUTMD<1,2> = "YES"  
OUTMD<1,3> = 4  
OUTMD<1,4> = "NO"  
RESULT = ""  
*  
CALL BMVR.DBASE.ENGINE(INF, INFLDS, XKEY, XVAL, OUTF, OUTMD, RESULT)
```

Figure 3

```
OSBWRITE DBF.DATA.REC ON DBF.FV AT DBF.BYTE  
DBF.BYTE += LEN(DBF.DATA.REC)  
DBF.DATA.REC = ""
```

Figure 4

	A	B	C	D	E	F
1	CUSTINFO	BEST BUY (007061)				
2	PACKTYPE	No-Tie				
3	FOLDTYPE	1/2				
4	PRESSID	13-7				
5						
6	Ship To	Ship To City	Ship To State	Ordered	Produced	Shipped
7	ALBANY DEMOCRAT-HERA	Albany	OR	41,577	41,675	41,675
8	BAINBRIDGE ISLAND RE	Everett	WA	6,480	6,495	6,495
9	BELLINGHAM HERALD (4	Mount Vernon	WA	62,475	62,600	62,600
10	BEND BULLETIN (19466	Bend	OR	76,311	76,665	61,640
11	BILLINGS GAZETTE (12	Billings	MT	79,797	79,910	79,910
12	BOISE IDAHO STATESMA	Boise	ID	194,298	193,714	193,714
13	CENTRAL KITSAP REPOR	Everett	WA	19,236	19,234	19,234
14	CENTRALIA DAILY CHRO	Centralia	WA	24,771	25,047	25,047
15	COEUR D'ALENE PRESS	Coeur D Alene	ID	72,000	72,114	72,114
16	CORVALLUS GAZETTE TI	Albany	OR	26,928	26,995	26,995
17	DO NOT USE (1831JC)	Anchorage	AK	78,318	78,420	78,420
18	DO NOT USE (485052)	Bremerton	WA	64,188	64,185	64,185
19	EUGENE REGISTER GUAR	Eugene	OR	145,113	145,015	145,015
20	EVERETT HERALD (4872	Everett	WA	144,000	144,345	144,345

Figure 5

Dashboard | Job Information | Price Matrix |

Billing Job-# 13-1993 Status INVOICED in AR / FINALIZED Refreshed 12/05/13 12:35PM  
Job Name ALBERTSON'S NW 12/11/13 Saved 01/15/14 05:47PM

Configuration Filter 01\*04 SACRAMENTO - CA [Appr: YES] per/M: YES  
02\*01 SACRAMENTO - CA [Appr: YES] per/M: YES

Code Filter ALL Sort Division + Code + Form  
Totals Configuration totals  
 Do NOT update line items from order.  
UPS Mode

INVOICE Line Items Filter Status Display FILTERED by Configs Display TOTAL \$78,710.36

Invoice Code	Invoice Code Description	Unit Price	Unit of Measure	Quantity	Total Price	Division	Form Cfg	Chg
066	COLOR PLATE CHANGES	\$60.00	EA	24	\$1,440.00	SACRAMENT(01-04	YES-z	
130	PLATE CHANGES	\$60.00	EA	64	\$3,840.00	SACRAMENT(02-01	YES-z	
252	FUEL SURCHARGE	\$925.48	EA	1	\$925.48	SACRAMENT(01-04	YES-z	
252	FUEL SURCHARGE	\$813.64	EA	1	\$813.64	SACRAMENT(02-01	YES-z	
255	FREIGHT DELIVERY STORE & OF	\$46.41	EA	1	\$46.41	SACRAMENT(01-04	YES-z	
256	U.P.S.	\$175.46	EA	1	\$175.46	SACRAMENT(01-04	YES-z	
257	FREIGHT	\$3.57	M	1,364.417	\$4,870.97	SACRAMENT(01-04	YES-z	
257	FREIGHT	\$3.57	M	1,199.536	\$4,282.34	SACRAMENT(02-01	YES-z	
311	OFF-LINE QUARTERFOLDING	\$2.50	M	2,703	\$6.76	SACRAMENT(01-04	YES-z	
471	PRINTING - NEWSPAPER INSER*	\$24.05	M	1,363.181	\$32,784.50	SACRAMENT(01-04	NO-z	
471	PRINTING - NEWSPAPER INSER*	\$24.05	M	1,199.536	\$28,848.84	SACRAMENT(02-01	NO-z	
474	PRINTING - MAILERS	\$24.05	M	136	\$3.27	SACRAMENT(01-04	NO-z	
475	PRINTING - STORE/OFFICE COPI	\$24.05	M	23,814	\$572.73	SACRAMENT(01-04	NO-z	
CSO	CARTONS - GENERAL PRINTING	\$1.75	EA	42	\$73.50	SACRAMENT(01-04	YES-z	
OVR	Overs	\$24.05	M	1,100	\$26.40	SACRAMENT(01-04	NO-z	

F1-Help F2-Save F3-Search F4-Exit F5-Filter F6-Edit F7-Refresh F8-Reports F9-Finalize

Figure 6

CHANGE. ADAPT. EVOLVE.

39<sup>TH</sup> Annual Conference

APRIL 20 – 23, 2020 | SADDLEBROOK RESORT, TAMPA, FL

Upon return you handle the delivery of the file. Clean-up the reporting file by removing all the indexed records you added for this instance of the report.

## Delivering Results to Users

The final .dbf file must be copied to a specific folder/file name defined in the ODBC configuration of your pivot table. The best results are obtained by configuring a folder on the root (C:\) drive of a user's desktop.

When processing an on-demand report request from a single user a simple way to make this happen is to send the .dbf via your email system. You can include both the .dbf and a copy of the associated Excel pivot table template as attachments to the email. The email message provides instructions for the user to detach the .dbf to the defined ODBC folder.

When processing a report generated by a phantom and intended to be accessed by any number of users, the .dbf can be written to a Windows network share. Macro programming in the pivot table template copies the file from the network share to the user's ODBC folder. The user can select from available .dbf versions. The pivot table template uses a defined file name in a defined location.

Note that the Excel pivot template can be refreshed against new data repeatedly while maintaining all formatting and calculations. The pivot table can have multiple pre-formatted tabs.

Figure 5 is a simple example of how **Basic MV Reports** delivers results. The pivot table itself can provide a significant amount of row-level calculation; data formatting and, of course, the slicing & dicing that pivot tables are designed for.

## MultiValue Data Handling

Handling multi-value arrays in your report requires that you write an individual report record for each element in the array. Build a base report record containing data for all single value elements you are using and use this as a base for each of the multi-value array elements.

Figure 6 is an SBclient user interface displaying a single U2 record using an MV array detailing the invoice line items for a billing job.

Converted to dBASE and displayed in Excel the results look like this. The source data could contain any number of U2 records with the MV arrays flattened. This allows the pivot table to fil-

ter and sort using any combination of data elements. For example, you might want to compare pricing across customers for one or more invoice codes [Figure 7].

## Setting Up ODBC Connections In Excel

In Windows 10, access the Control Panel; double-click Administrative Tools; double-click ODBC Data Sources (32-bit) to display this dialog [Figure 8].

Configure as shown by selecting dBase IV and the directory where you are storing the .dbf files.

Open a new Excel workbook and Insert a pivot table. Select “Use an exter-

nal data source” and click on “Choose Connection”. In the next dialog click on “Browse for More ...” and then click on “New Source”. Select OBDC DSN from the list and click Next. Choose the OBDC data source that you created above. Finally select the .dbf file from the displayed list and click Next to add a description of this connection and click Finish. Click OK on the final dialog box. You can now begin to format your pivot table. After formatting and saving the pivot table you can use this workbook again each time you update the .dbf table that you pointed it at.

You may already know about OBDC data sources. For more information you can check this website or any of the numerous sources on the internet.

<https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2018/ENU/AutoCAD-Customization/files/GUID-A7842E65-0BF1-4D41-9CCA-05AFA5AACF10.htm.html> IS

A	B	C	D	E
1 Region	WEST			
2 Billing Division	PORLAND - OR			
3 Job Name	ALBERTSON'S NW 12/11/13			
4 Customer	IVIE & ASSOCIATES, INC. - PORTLAND (1066)			
5 Prod Division	SACRAMENTO - CA			
6 Press	(All)			
7 Invoice Category	(All)			
8				
10 Config-ID	Line Item	Price	Quantity	Extension
11 13-3993-01-04	PRINTING - MAILERS (474)	\$24.05	136	\$3.26
12	PRINTING - STORE/OFFICE COPIES (475)	\$24.05	23,814	\$572.72
13	PRINTING - NEWSPAPER INSERTS (471)	\$24.05	1,363,181	\$32,784.50
14	Overs (OVR)	\$24.05	1,100	\$26.45
15	OFF-LINE QUARTERFOLDING (311)	\$2.50	2,703	\$6.76
16	U.P.S. (256)	\$175.46	1	\$175.46
17	CARTONS - GENERAL PRINTING (CSO)	\$1.75	42	\$73.50
18	COLOR PLATE CHANGES (066)	\$60.00	24	\$1,440.00
19	FREIGHT (257)	\$3.57	1,364,417	\$4,870.98
20	FUEL SURCHARGE (252)	\$231.37	4	\$925.48
21	FREIGHT DELIVERY STORE & OFFICE COPIES (	\$46.41	1	\$46.41
22				
23 13-3993-02-01	PRINTING - NEWSPAPER INSERTS (471)	\$24.05	1,199,536	\$28,848.84
24	FREIGHT (257)	\$3.57	1,199,536	\$4,282.35
25	FUEL SURCHARGE (252)	\$90.40	9	\$813.64
26	PLATE CHANGES (130)	\$60.00	64	\$3,840.00
27				

Figure 7

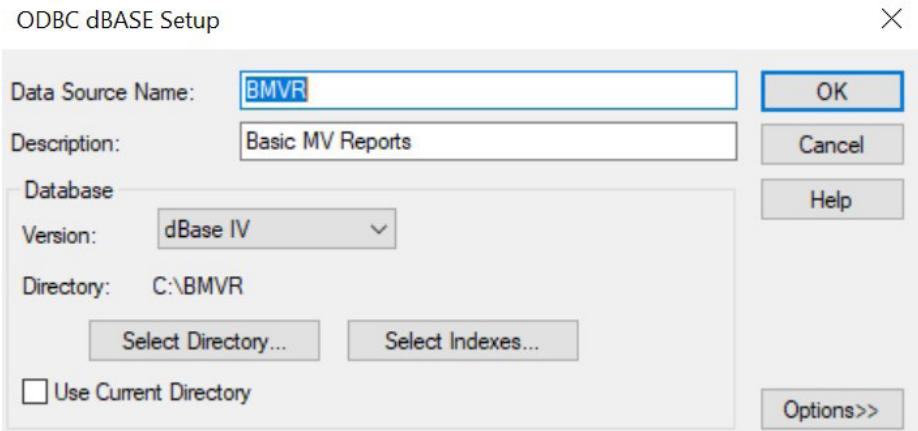


Figure 8

## Feedback

*What came first, the letters or the letters-to-the-editor department?*

International Spectrum Magazine has a Feedback Department, sometimes known as Letters to the Editor.

We want to hear your comments, your reactions, your agreement or disagreement with what you see. Also, do not hesitate to let us know about things happening in the MultiValue Community we may not have heard about yet.

Please send your comments by e-mail to:  
editor@intl-spectrum.com

# MARKETPLACE

## ACCOUNTING

**Natec Systems**  
www.natecsystems.com | nrector@natecsystems.com

## COMPLIANCE

**SJ+ Systems Associates**  
www.sjplus.com | sjoslyn@sjplus.com

## CONSULTING

**Drexel Management Service**  
www.drexelmgt.com | dconboy@drexelmgt.com

**Execu-Sys, LTD**  
www.eslny.com | mh@eslny.com

**HDWP**  
www.HDWP.com | results@HDWP.com

**Modern MultiValue, LLC**  
www.ModernMultiValue.com | info@ModernMultiValue.com

**PICK Programmers Shop**  
www.pickprogram.com | brian@pickprogram.com

**Precision Solutions**  
www.precisionline.com | Kevin@PrecisOnline.com

## DATABASE

**Zumasys**  
www.zumasys.com/products/accuterm/

## FILE MANAGEMENT

**Paradigm Systems, Inc.**  
www.paradigm-systems.us | sales@paradigm-systems.us

## REPORTING

**Brian Leach Consulting, LTD**  
www.brianleach.co.uk | brian@brianleach.co.uk

## TERMINAL EMULATOR

**Zumasys**  
www.zumasys.com/products/accuterm/



The AccuTerm logo features a stylized 'A' icon composed of a red triangle pointing up and a white 'A' shape. To the right of the icon, the word "AccuTerm" is written in a bold, white, sans-serif font, with a registered trademark symbol (®) at the end.

**The most popular terminal emulator for PICK just got better with a web interface and dozens of exciting new features. AccuTerm 8 is here!**

Learn more: [zumasys.com/accuterm](http://zumasys.com/accuterm)

## WEB DEVELOPMENT AND TOOLS

**Aptron Corporation**  
www.aptron.com | info@aptron.com

## LETTERS TO THE EDITOR

Have an opinion on an article: Agree, disagree, or enhancement to an article from a previous issue? International Spectrum and our authors are interested in hearing from you!

E-mail: editor@intl-spectrum.com

## WANT TO SEE A SPECIFIC TOPIC?

International Spectrum is looking for writers, feedback, and topic ideas. We all have specific topics and issues that we need answers to find solutions for. Send us an E-mail with topics you would like to have covered in the magazine or on the website.

E-mail: nathan@intl-spectrum.com

## WANT TO WRITE?

Expand your professional credentials, and provide us with an article.

Give us a rough and ugly outline, and we will help you refine it, proof it, and make it press ready. Or you can give us something polished, proofed, and press ready to publish.

Share your thoughts and expertise with over 10,000 fellow MultiValue developers and users.

E-mail: editor@intl-spectrum.com

## NEED A MENTOR?

Mentors give developers the ability to ask industry experts for direction, code examples, and/or just ask them to see if something makes sense. Sometimes, all you need is a resource or example to start or complete a project.

Check with us to see who is available for mentoring, and how you can take advantage of it to save your business or company money.

E-mail: nathan@intl-spectrum.com

## WANT TO BE A MENTOR?

We have many retired or semi-retired professionals out there that would love to share their knowledge of MultiValue development. If you are one of them, please contact us to see what mentoring is all about.

E-mail: nathan@intl-spectrum.com

# The Rosetta Stone Project

BY CHARLES BAROUCH,  
WITH ADDITIONAL CODE BY  
AARON YOUNG AND DICK THIOT

**W**hen we work in PHP, or Python, or Ruby, or C#, we have certain tools.

When we work in MultiValue, we have certain tools. I'm greedy, I want the best of both. This is part one in a series on how to create MultiValue features in PHP, Python, Ruby, Node.js, and C#.

Not only does this give us a taste of MV in the other languages we pick up, but it also becomes a sort of Rosetta Stone. For those who don't know the term, the Rosetta Stone was a tablet with three written languages on it, each declaring the same information. Having them side-by-side meant that if you could read any one of the language, you could use it as a guide toward learning the other two.

So, seeing code in C# or Ruby means that for people who know one of them, the parallels can help programmers bridge from one into the other. Hat tip to my cousin Alana for suggesting the Rosetta Stone metaphor.

---

*So, seeing code in C# or Ruby... can help programmers bridge from one into the other.*

---

To get this started, we've elected to implement some of the nicer features of OCONV.

## DATE

We won't be implementing every variation of the Date Conversion but we will implement the core features. And, since you'll have the source code, so you can expand it. We'll get the "/" vs. "-" working [Figure 1].

## TIME

As with Date, above, we'll just implement a subset of the features [Figure 2].

## GROUP EXTRACT

Our Group Extract is slightly more robust than the MultiValue version. It will accept multiple character delimiters. Included in the code is notes on

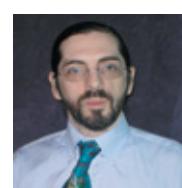
how to scale it back if you want single characters only [Figure 3].

## EXAMPLES OF USE

See Figure 4.

Next article, we'll build on these examples and talk more about how to use them to reach Ruby, C#, Python, Node.js, and PHP people how MultiValue works. By then, we may have a few more languages added.

You can find this code on GitHub: [https://github.com/CharlesBarouch/mv\\_core](https://github.com/CharlesBarouch/mv_core). You can also create a branch and start adding features and corrections. We welcome your participation. **IS**



CHARLES BAROUCH is the CTO of HDWP, Inc. and the Publisher at HDWPbooks. You can read his writing in International Spectrum, Theme-Thology, Novo Pulp, Pax Solaria, PerehelionSF, and the Interrogative series, which begins with Tiago and the Masterless.

```

PHP Code (Full code: https://github.com/CharlesBarouch/mv\_core)
<?php
// mv_core.php
// by Charles Barouch (Results@HDWP.com)
// on 09/15/19
// Originally published in Intl-Spectrum.com
// -----
function mv_oconv($value,$rule)
{
    $rule_1 = strtoupper(substr($rule,0,1));
    $rule_2 = substr($rule,0,2);
    if(      $rule_1 == 'D') { $result = mv_oconv_date($value,$rule);}
    else if($rule_1 == 'G') { $result = mv_oconv_group($value,$rule);}
    else if($rule_2 == 'MT') { $result = mv_oconv_time($value,$rule);}
    return $result;
}

function mv_oconv_date($value,$rule)
{
    $dt = new DateTime("1967-12-31 00:00:00");
    $dt->Modify('+'.$value.' day');
    $mdy = [$dt->format("m"),$dt->format("d"),$dt->format("Y")];
    $dlmtr = '/';
    if(strpos($rule,'-',1)) {$dlmtr = '-';}
    if(is_numeric(substr($rule,1,1)))
    {
        $mdy[2] = substr($mdy[2],-1*substr($rule,1,1));
    }
    if(strpos($rule,'Y')) {$result = $mdy[2];} else {
        if(substr($rule,0,2) == 'DM') {$result = $mdy[0];} else {
            if(substr($rule,0,2) == 'DD') {$result = $mdy[1];} else {
                $result = $mdy[0] . $dlmtr . $mdy[1] . $dlmtr . $mdy[2];
            }
        }
    }
    return $result;
}
?>

PYTHON (Full code: https://github.com/CharlesBarouch/mv\_core)
import datetime
# mv_core.py
# by Charles Barouch
# on 09/15/19
# Originally published in Intl-Spectrum.com
# -----

def oconv_date(value, rule):
    baseline = datetime.date(1967, 12, 31)
    result = baseline + datetime.timedelta(days=value)
    # Digits in a year
    YearStart = 0
    YearFinish = 4
    if("2" in rule):
        YearStart = 2
        YearFinish = 4
    delimiter = '/'
    if("-" in rule):
        delimiter = "-"
    if("Y" in rule):
        result = str(result.year)[YearStart:YearFinish]
    else:
        if("DM" in rule):
            result = str(result.month)
        else:
            if("DD" in rule):
                result = str(result.day)
            else:
                result = str(result.month) + delimiter + str(result.day) + delimiter + (str(result.year))

```

**Figure 1**

```

[YearStart:YearFinish])
    return result

def oconv(value, rule):
    rule = rule.upper()
    if rule[0] == 'D':
        result = oconv_date(value,rule)
    if rule[0] == 'G':
        result = oconv_group(value,rule)
    if rule[0:2] == 'MT':
        result = oconv_time(value,rule)
    return result

RUBY (Full code: https://github.com/CharlesBarouch/mv\_core)
# mv_core.rb
# by Aaron Young (brainomite@gmail.com)
# on 09/30/19
# ----

require "Date"

def mv_oconv(value, rule)
    upcased_rule = rule.upcase # ensure rule is uppercase
    one_letter_rule = upcased_rule[0]
    two_letter_rule = upcased_rule[0..1] # get the first two letters using a range
    if one_letter_rule == "D" # its a date
        result = mv_oconv_date(value.to_i, upcased_rule)
    elsif one_letter_rule == "G" # its a group
        result = mv_oconv_group(value, upcased_rule)
    elsif two_letter_rule == "MT" # its a time
        result = mv_oconv_time(value.to_i, upcased_rule)
    else
        result = nil
    end
    result.to_s
end

def mv_oconv_date(value, rule)
    # create a date starting from 12/31/1967 and add value (days) to it
    date = Date.new(1967,12,31) + value

    case rule
    when "DM"
        date.strftime("%m") # zero padded month string
    when "DD"
        date.strftime("%d") # zero padded day string

    # regular expression for a full date with delimiters i.e. "D2-"
    when /D[1234][-\/]/
        get_date(date, rule)

    # regular expression for year with a length i.e. "D4Y"
    when /D[1234]Y/
        get_year(date, rule)
    end
end

NODEJS (Full code: https://github.com/CharlesBarouch/mv\_core)
// mvCore.js
// by Aaron Young (brainomite@gmail.com)
// on 10/13/19
// ----

const mvOconv = (value, rule) => {
    upcasedRule = rule.toUpperCase();

```

**Figure 1 Contined**

```

oneLetterRule = upcasedRule[0];
twoLetterRule = upcasedRule.substring(0, 2);

if (oneLetterRule === "D") {
  return mvOconvDate(Number.parseInt(value), upcasedRule);
} else if (twoLetterRule === "MT") {
  return mvOconvTime(Number.parseInt(value), upcasedRule);
} else if (oneLetterRule === "G") {
  return mvOconvGroup(value, upcasedRule);
}
};

const mvOconvDate = (value, rule) => {
  const mvEpoch = new Date(1967, 11, 31);
  let date = mvEpoch.addDays(value);

  if (/D[1234][-\/]/.test(rule)) {
    // regular expression for a full date with delimiters i.e. "D2-"
    return getDate(date, rule);
  } else if (/D[1234]Y/.test(rule)) {
    // regular expression for year with a length i.e. "D4Y"
    return getYear(date, rule);
  } else if (rule === "DM") {
    return pad(date.getMonth() + 1); // zero based months
  } else if (rule === "DD") {
    return pad(date.getDate());
  } else {
    return "oops";
  }
};

const getYear = (date, rule) => {
  years = date.getFullYear().toString();
  chars = Number.parseInt(rule[1]);
  return years.substring(4 - chars);
};

const getDate = (date, rule) => {
  day = pad(date.getDate());
  month = pad(date.getMonth() + 1); // zero-based months need to add 1
  year = getYear(date, rule);
  delim = rule[2];
  return `${month}${delim}${day}${delim}${year}`;
  // return "yay";
};

// helpers

const isInteger = string => Number.isInteger(Number.parseInt(string));

const pad = number => {
  if (number < 10) {
    return "0" + number;
  }
  return number.toString();
};

Date.prototype.addDays = function(days) {
  // https://stackoverflow.com/questions/563406/add-days-to-javascript-date
  var date = new Date(this.valueOf());
  date.setDate(date.getDate() + days);
  return date;
};

const findFirstNonNumericValue = value => {
  for (char of value) {
    if (!isInteger(char)) {
      return char;
    }
  }
};

```

**Figure 1 Contined**

```

    }

};

module.exports = {
  mvOconv
};

C# (Full code: https://github.com/CharlesBarouch/mv\_core)
using System;

using System.Globalization;

namespace mv_core
{
  public class mv_conv
  {
    public string mv_oconv(string value, string rule)
    {
      string result = "";
      string rule1 = rule.ToUpper().Substring(0, 1);
      string rule2 = rule.Substring(0, 2);
      if (rule1 == "D")
      {
        result = mv_oconv_date(value, rule);
      }
      else if (rule2 == "MT")
      {
        result = mv_oconv_time(value, rule);
      }
      else if (rule1 == "G")
      {
        result = mv_oconv_general(value, rule);
      }
      else { result = ""; }

      return result;
    }
    private string mv_oconv_date(string value, string rule)
    {
      string result = "";
      DateTime dt = new DateTime(1967, 12, 31);
      dt = dt.AddDays(Convert.ToInt16(value));

      //break into elements
      string dt_day = dt.ToString("dd");
      string dt_mo = dt.ToString("MM");
      string dt_yr = dt.ToString("yyyy");

      string dt_day_shortname = dt.ToString("ddd");
      string dt_mo_shortname = dt.ToString("MMM");
      string dt_yr2 = dt_yr.Substring(2, 2);

      string separator = rule.Contains("/") ? "/" : rule.Contains("-") ? "-" : " ";
      string toReturn = string.Concat("{0}", separator, "{1}", separator, "{2}");

      switch (rule)
      {
        case "D2/":
        case "D2-":
          result = String.Format(toReturn, dt_mo, dt_day, dt_yr2);
          break;
        case "D2":
        case "D4":
          result = String.Format(toReturn, dt_day, dt_mo_shortname,

```

**Figure 1 Contined**

```

(rule == "D2" ? dt_yr2 : dt_yr));
        break;
    case "D4/":
    case "D4-":
        result = String.Format(toReturn, dt_mo, dt_day, dt_yr);
        break;
    case "DD":
        result = dt_day;
        break;
    case "DW":
        result = (Convert.ToInt32(dt.DayOfWeek) * 1).ToString();
        break;
    case "DWA":
        result = dt.ToString("dddd");
        break;
    case "DWB":
        result = dt_day_shortname;
        break;
    case "DM":
        result = dt_mo;
        break;
    case "DMA":
        result = dt.ToString("MMMM");
        break;
    case "DMB":
        result = dt_mo_shortname;
        break;
    case "DQ":
        result = GetQuarter(dt).ToString();
        break;
    case "DY":
        result = dt_yr;
        break;
    case "DY2":
        result = dt_yr2;
        break;
    case "DY4":
        result = dt_yr;
        break;
    }
    return result.ToUpper();
}
}

```

**Figure 1 Continued**

```

PHP
function mv_oconv_time($value,$rule)
{
    $hour = floor($value / 3600);
    $minute = floor(($value - $hour*3600)/ 60);
    $second = $value - ($hour*3600 + $minute*60);
    $apm = '';
    if (substr($rule,2,1) == 'H')
    {
        $hour = ($hour % 24);
        if($hour >= '00' && $hour <= '11') {$apm = 'am';} else {$apm = 'pm'; $hour = $hour - 12;}
    }
    $hour = str_pad($hour, 2, "0", STR_PAD_LEFT );
    $minute = str_pad($minute, 2, "0", STR_PAD_LEFT );
    $second = str_pad($second, 2, "0", STR_PAD_LEFT );
    $result = $hour . ':' . $minute . ':' . $second . $apm;
    return $result;
}

PYTHON
def oconv_time(value,rule):
    result = datetime.timedelta(seconds=value)
    return str(result)

```

**Figure 2**

```

RUBY
def mv_oconv_time(value, rule)
  time = Time.at(value) # create a time object using seconds
  time.gmtime # remove utc offsets so it isn't skewed
  # convert to a string
  if rule == "MTS" # use military time
    time.strftime("%H:%M:%S")
  elsif rule == "MTHS" # use non-military time with a meridiem indicator
    time.strftime("%I:%M:%S%P")
  else
    nil # return nothing, not a valid rule
  end
end

NODEJS
const mvOconvTime = (value, rule) => {
  const time = new Date(value * 1000); // uses miliseconds
  const seconds = pad(time.getUTCSeconds());
  const minutes = pad(time.getUTCMilliseconds());
  if (rule === "MTS") {
    const hours = pad(time.getUTCHours());
    return `${hours}:${minutes}:${seconds}`;
  }
  if (rule === "MTHS") {
    let hours;
    const utcHours = time.getUTCHours();
    if (utcHours === 0) {
      hours = 12;
    } else if (utcHours > 12) {
      hours = pad(utcHours - 12);
    } else {
      hours = pad(utcHours);
    }
    const AMorPM = utcHours < 12 ? "AM" : "PM";
    return `${hours}:${minutes}:${seconds}${AMorPM}`;
  }
};

C#
private string mv_oconv_time(string value, string rule)
{
    string result = "";
    Int32 value_time = Convert.ToInt32(value);
    Int32 hour = (value_time / 3600);
    Int32 minute = ((value_time - (hour * 3600)) / 60);
    Int32 second = ((value_time - (hour * 3600) + (minute * 60)));
}

```

**Figure 2 Continued**

```

PHP
function mv_oconv_group($value,$rule)
{
  // Split up the Rule into Skip, Delimiter, and Take
  $skip = 0;
  $take = 0;
  $dlmtr = '';
  $rpos = 0;
  $smax = strlen($rule);
  for ($scnt = 1; $scnt < $smax; $scnt++)
  {
    $chr = $rule[$scnt];
    if(is_numeric($chr))
    {
      if($rpos == 0){$skip .= $chr;} else {$take .= $chr;}
    } else {
      if($dlmtr == ''){ $dlmtr = $chr; }
      $rpos = 2;
    }
  }
}

```

**Figure 3**

```

        }
    }
$result = '';
$temp = explode($dlmtr,$value);
$skip += 0; // Force numeric
$rmax = $skip + $take;
for($rcnt = $skip; $rcnt < $rmax; $rcnt++)
{
    if($result != '') { $result .= $dlmtr;}
    $result .= $temp[$rcnt];
}
return $result;
}

```

#### PYTHON

```

def oconv_group(value,rule):
    # split rule into skip, delimiter, and take
    skip = 1
    take = 3
    delimiter = '!'
    # apply rule
    result = ''
    value = value.split(delimiter)
    for parts in value:
        if skip > 0:
            skip -= 1
        else:
            if take > 0:
                take -= 1
                if result != '':
                    result += delimiter
                result += parts
    return result

```

#### RUBY

```

def mv_oconv_group(value, rule)
    actual_rule = rule[1..-1] # remove first char
    delimiter = find_first_non_numeric_value(actual_rule) # find the delimiter

    # take the rule and turn into an array using the delimiter then
    # convert all elements into integers and assign the first
    # value to skip_num and second value to take_num
    skip_num, take_num = actual_rule.split(delimiter).map(&:to_i)
    array = value.split(delimiter) # create an array using the delimiter

    # create a sub array by skipping skip_num numbers then take the first
    # take_num elements and return the new resulting array
    array[skip_num..-1].take(take_num)
end

```

#### NODEJS

```

const mvOconvGroup = (value, rule) => {
    const actualRule = rule.substring(1);
    const delimiter = findFirstNonNumericValue(actualRule);
    const [skip_num, take_num] = actualRule
        .split(delimiter)
        .map(val => Number.parseInt(val));
    const fullArray = value.split(delimiter);
    const subArray = fullArray.slice(skip_num);
    const resultArray = subArray.slice(0, take_num);
    return resultArray.toString();
};

```

#### C#

Forthcoming

**Figure 3 Continued**

```

PHP
<?php
// add the function to this script
include_once('..../mv_core.php');
//
// Load Test Cases
$stack = file_get_contents('..../teststack.txt');
$stack = explode('^', $stack);
echo 'Loaded Test Cases' . "\r\n";
foreach($stack as $testcase)
{
    $testcase = explode(']', $testcase);
    echo mv_oconv($testcase[0], $testcase[1]) . "\r\n";
}
//
// Run some pre-set cases
echo "\r\n";
echo 'Hardcoded Cases' . "\r\n";
echo mv_oconv(-1200, 'D2/') . "\r\n";
echo mv_oconv(18500, 'D2/') . "\r\n";
echo mv_oconv(18500, 'D4-') . "\r\n";
echo mv_oconv(18500, 'DM') . "\r\n";
echo mv_oconv(18500, 'DD') . "\r\n";
echo mv_oconv(18500, 'D2Y') . "\r\n";
echo mv_oconv(86375, 'MTS') . "\r\n";
echo mv_oconv(86375, 'MTHS') . "\r\n";
echo mv_oconv('A!BB!CCC!DDD!DDD', 'G1!3');
?>

```

#### **PYTHON**

```

import mv_core as mv

print(mv.oconv(18575, 'D2/'))
print(mv.oconv(18575, 'D4-'))
print(mv.oconv(18575, 'DM'))
print(mv.oconv(18575, 'DD'))
print(mv.oconv(18575, 'D2Y'))
print(mv.oconv(86375, 'MTS'))
print(mv.oconv(86375, 'MTHS'))
print(mv.oconv('A!BB!CCC!DDD!DDD', 'G1!3'))

```

#### **RUBY**

```

# test.rb
# by Aaron Young (brainomite@gmail.com)
# on 09/30/19
# -----
require_relative "mv_core.rb"

puts "Loaded Test Cases"
file = File.open(__dir__ + "..../teststack.txt")
# read the file and remove linefeeds
stack_data = file_data = file.read.chomp
tests = stack_data.split("^")
tests.each do |test|
    params = test.split("]")
    value = params[0]
    rule = params[1]
    expected = params[2]
    puts "mv_oconv(#{value}, \"#{rule}\") - Expected: '#{expected}' - Actual: '#{mv_oconv(value, rule)}'"
end
puts ""

# Run some pre-set cases
puts "Hardcoded Cases"
puts mv_oconv(18500, 'D2/') # 08/25/18

```

**Figure 4**

```

puts mv_oconv(18500,'D4-') # 08-25-2018
puts mv_oconv(18500,'DM') # 08
puts mv_oconv(18500,'DD') # 25
puts mv_oconv(18500,'D2Y') # 18
puts mv_oconv(86375,'MTS') # 23:59:35
puts mv_oconv(86375,'MTHS') # 11:59:35pm
puts mv_oconv('A!BB!CCC!DDD!DDD','G1!3') # ["BB", "CCC", "DDD"]

NODEJS
// by Aaron Young (brainomite@gmail.com)
// on 09/30/19
// -----
const { mvOconv } = require("./mvCore");
const path = require("path");

const filePath = path.join(__dirname, "..", "teststack.txt");
console.log("Loaded Test Cases");
const stackData = require("fs")
  .readFileSync(filePath, "utf-8")
  .split("\n")
  .filter(Boolean)[0]; // get first line sans new line chars
tests = stackData.split("^");
for (test of tests) {
  const [value, rule, expected] = test.split("]");
  const result = mvOconv(value, rule);
  console.log(`mv_oconv(${value}, ${rule}) - Expected: '${expected}' - Actual: '${result}'`)
}

console.log("");
console.log("Hardcoded Cases");
console.log(mvOconv(18500, "D2/")); // 08/25/18
console.log(mvOconv(18500, "D4-")); // 08-25-2018
console.log(mvOconv(18500, "DM")); // 08
console.log(mvOconv(18500, "DD")); // 25
console.log(mvOconv(18500, "D2Y")); // 18
console.log(mvOconv(86375, "MTS")); // 23:59:35
console.log(mvOconv(86375, "MTHS")); // 11:59:35PM
console.log(mvOconv("A!BB!CCC!DDD!DDD", "G1!3")); // BB,CCC,DDD

C#
using System;
using mv_core;

namespace mv_oconv_test
{
    class Program
    {
        static void Main(string[] args)
        {
            string test_date = "18915";
            var conv = new mv_core.mv_conv();
            string oconv_date = conv.mv_oconv(test_date, "D2/");
            Console.WriteLine("D2/ - " + oconv_date);
            oconv_date = conv.mv_oconv(test_date, "D2-");
            Console.WriteLine("D2- - " + oconv_date);
            oconv_date = conv.mv_oconv(test_date, "D2");
            Console.WriteLine("D2 - " + oconv_date);
            oconv_date = conv.mv_oconv(test_date, "D4/");
            Console.WriteLine("D4/ - " + oconv_date);
            oconv_date = conv.mv_oconv(test_date, "D4-");
            Console.WriteLine("D4- - " + oconv_date);
            oconv_date = conv.mv_oconv(test_date, "D4");
            Console.WriteLine("D4 - " + oconv_date);
            oconv_date = conv.mv_oconv(test_date, "DD");
        }
    }
}

```

**Figure 4 Contined**

```

Console.WriteLine("DD - " + oconv_date);
oconv_date = conv.mv_oconv(test_date, "DW");
Console.WriteLine("DW - " + oconv_date);
oconv_date = conv.mv_oconv(test_date, "DWA");
Console.WriteLine("DWA - " + oconv_date);
oconv_date = conv.mv_oconv(test_date, "DWB");
Console.WriteLine("DWB - " + oconv_date);
oconv_date = conv.mv_oconv(test_date, "DM");
Console.WriteLine("DM - " + oconv_date);
oconv_date = conv.mv_oconv(test_date, "DMA");
Console.WriteLine("DMA - " + oconv_date);
oconv_date = conv.mv_oconv(test_date, "DMB");
Console.WriteLine("DMB - " + oconv_date);
oconv_date = conv.mv_oconv(test_date, "DQ");
Console.WriteLine("DQ - " + oconv_date);
oconv_date = conv.mv_oconv(test_date, "DY");
Console.WriteLine("DY - " + oconv_date);
oconv_date = conv.mv_oconv(test_date, "DY2");
Console.WriteLine("Dy2 - " + oconv_date);
oconv_date = conv.mv_oconv(test_date, "DY4");
Console.WriteLine("DY4 - " + oconv_date);

string test_time = "12519";
string oconv_time = conv.mv_oconv(test_time, "MT");
Console.WriteLine("MTS - " + oconv_time);
oconv_time = conv.mv_oconv(test_time, "MTS");
Console.WriteLine("MT - " + oconv_time);
oconv_time = conv.mv_oconv(test_time, "MTHS");
Console.WriteLine("MTHS- " + oconv_time);

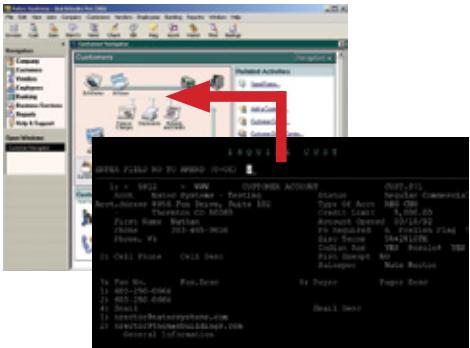
var ans = Console.ReadLine();
}
}
}

```

**Figure 4 Continued**

mv  
QB

## QuickBooks API for the MultiValue Database



- **Read/Write Directly to Quickbooks Databases**  
Customer, Vendor, Invoices, Purchase Orders, Chart of Accounts
- **mvQB API is Designed for the MultiValue Program to Use**  
All routines are simple BASIC calls designed for the developer. No special user interfaces required.
- **No Need to Learn the Internals of QuickBooks**
- **QuickBooks Pro/Premier/Enterprise**

**NATEC Systems** 

Providing Solutions to your MultiValue Questions

**Phone:** 303.465.9616  
**E-mail:** mvqb@natecsystems.com  
**Website:** www.natecsystems.com