SNAP PAC RESTful API to Access Database

About this Tutorial

The SNAP PAC RESTful API to Access Database Example is an interactive database that builds commands to read data from an Opto 22 SNAP PAC programmable automation controller and store the data in a Microsoft Access table.

This technical note describes how the example works.

If you have not already done so, download the SNAP PAC RESTful API to Access Database Example on our website at www.opto22.com. The download includes these files:

- Opto22_RESTfulPAC_Access.accdb—An Access database containing a data input form, two tables, and Visual Basic code. This database requires only the name or IP address of a PAC and the login and password for a user with Read privileges. (See "Prerequisite: Setting Up the PAC" on page 2.
- JsonConverter.bas—A third-party class for VBA created by MIT (copyright Tim Hall, https://github.com/VBA-tools/VBA-JSON). This file is available for general use as long as the provisions in its copyright are preserved.
- 2204_SNAP_PAC_RESTful_API_to_Access_Database_Technical Note.pdf—(This document)

The first part of this tech note explains the sample database. The optional second part of the tech note explains how to see and edit the code behind the sample, if you wish to.

Questions about this tutorial can be addressed to Support@opto22.com.

About the RESTful API

The REST API is a way for software applications to securely access data in an Opto 22 SNAP PAC controller by making GET and POST requests. The complete Opto 22 RESTful API is documented here:

http://developer.opto22.com/static/generated/pac-rest-api/swagger-ui/index.html

At this link you'll find URL paths to read any data available to the PAC Control strategy running on the PAC and to write to the same data, if it is a writable data type. All the GET and POST requests return information in standard JavaScript Object Notation (JSON).

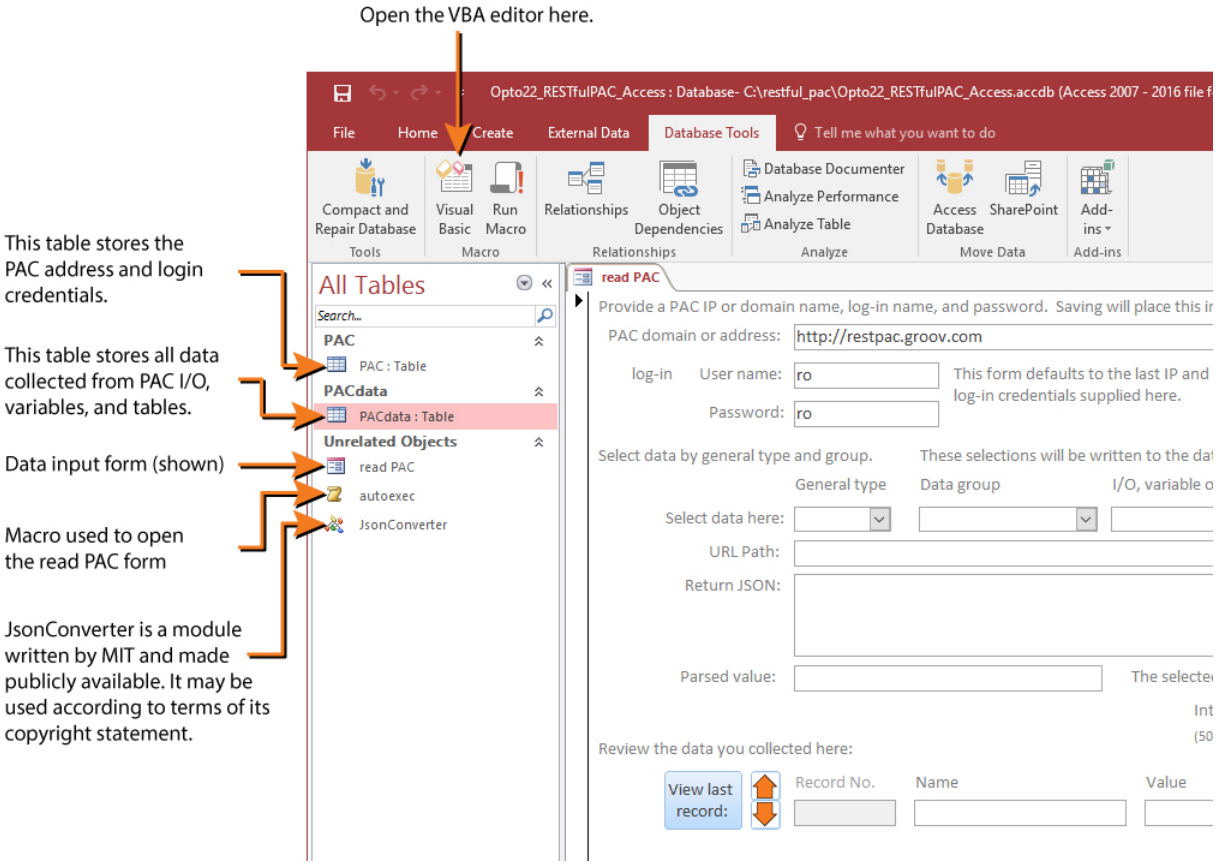
The Access sample file uses the PAC's RESTful API and common Access features to create an interactive database that can query your PAC and then build URL paths for the data in your control strategy. You don't need to know the I/O point and variable names on your PAC, because the Access sample can read the I/O and variable names and will create the paths. To see how to construct the HTTP requests and parse the returned data, you'll look in the Visual Basic code included in the Access sample.

Prerequisite: Setting Up the PAC

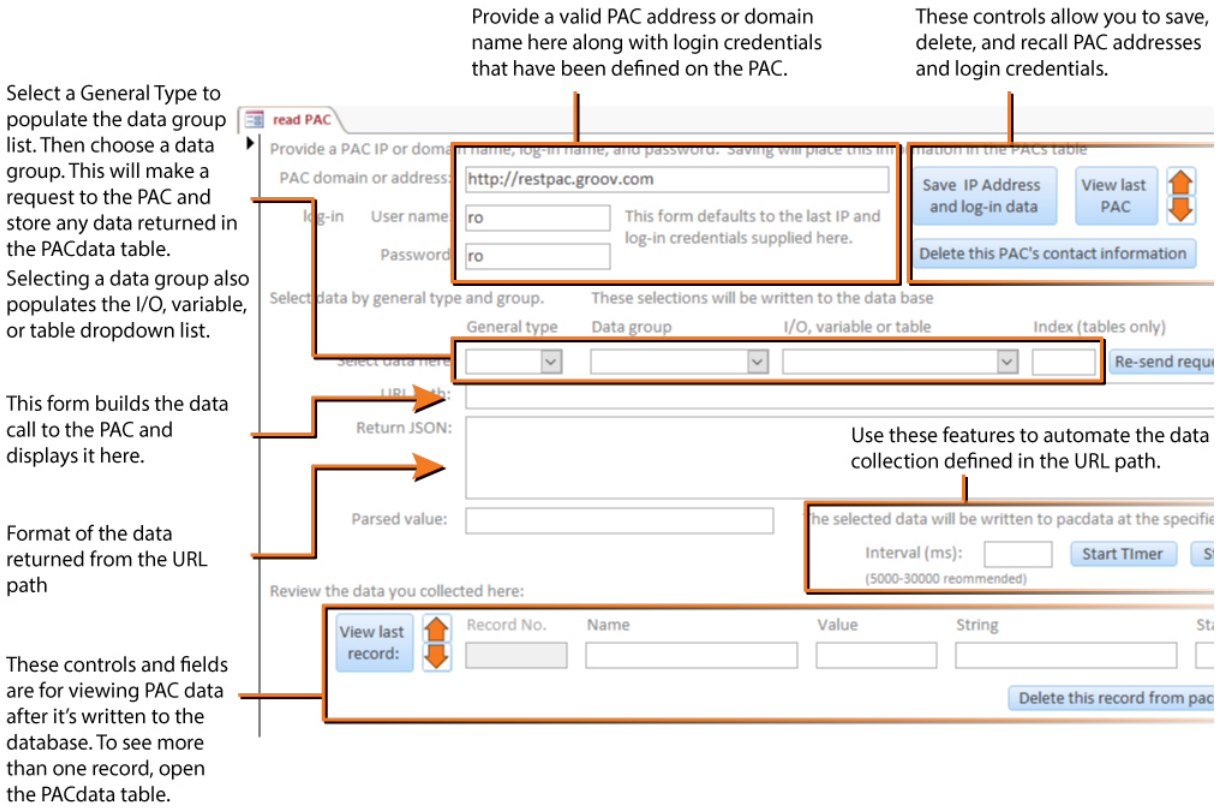
Before you begin, enable the RESTful interface on the PAC and create a user account with read privileges as described in the SNAP PAC REST API Quick Start, which is available at http://developer.opto22.com/rest/pac/quickstart/

About the Sample File

Open Access



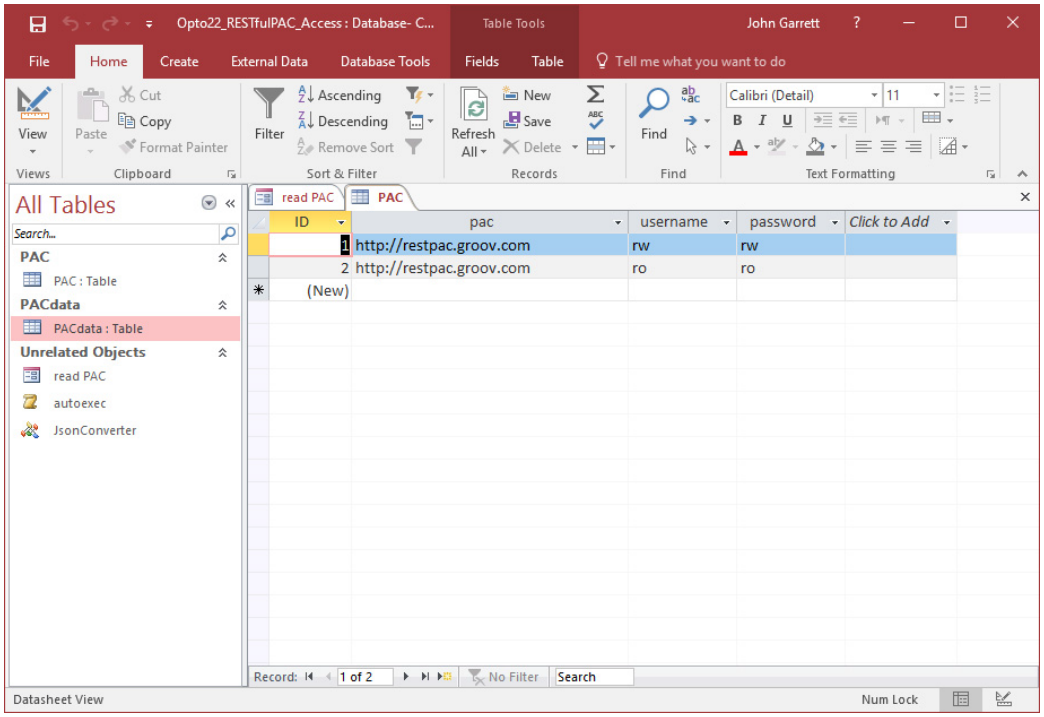
The read PAC form



The PAC table

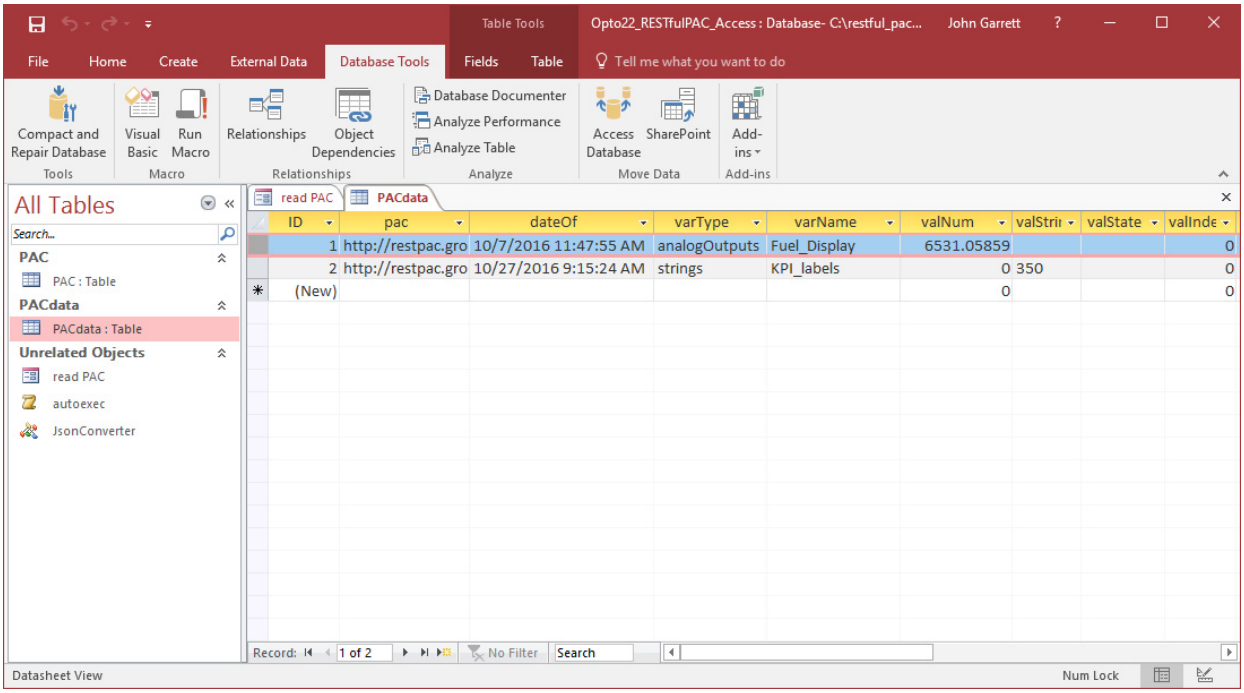
The PAC table holds SNAP PAC controller addresses or domain names and user credentials (usernames and passwords). To use the Access example, you don't have to save PAC addresses and user credentials to the PACs table; the table is provided for convenience. You can manage records from the read PAC form.

The PAC table is shown here with two sample login credentials for Opto 22's online REST demonstration.



The PACdata table

The data collected from the PAC's I/O, variables, and tables are stored in one table called PACdata. Data from the PAC are intended for numerical, Boolean, and string formats. The Visual Basic code uses the data group to determine whether a value is written to the valNum, valState, or valString fields. The data group name is stored with each record, so data can be retrieved by group as well as by name, date, PAC address, and so on.



The fields in the PACdata table are integers, strings, dates, and floating point numbers corresponding to "Double" in Access. Note that the data type of varNum is Double and set to include 8 decimal places, to preserve the floating point precision returned from the PAC.

NOTE: This database is for demonstration and is not intended to show the ideal organization for any specific real-world application.

Visual Basic Behind the Form

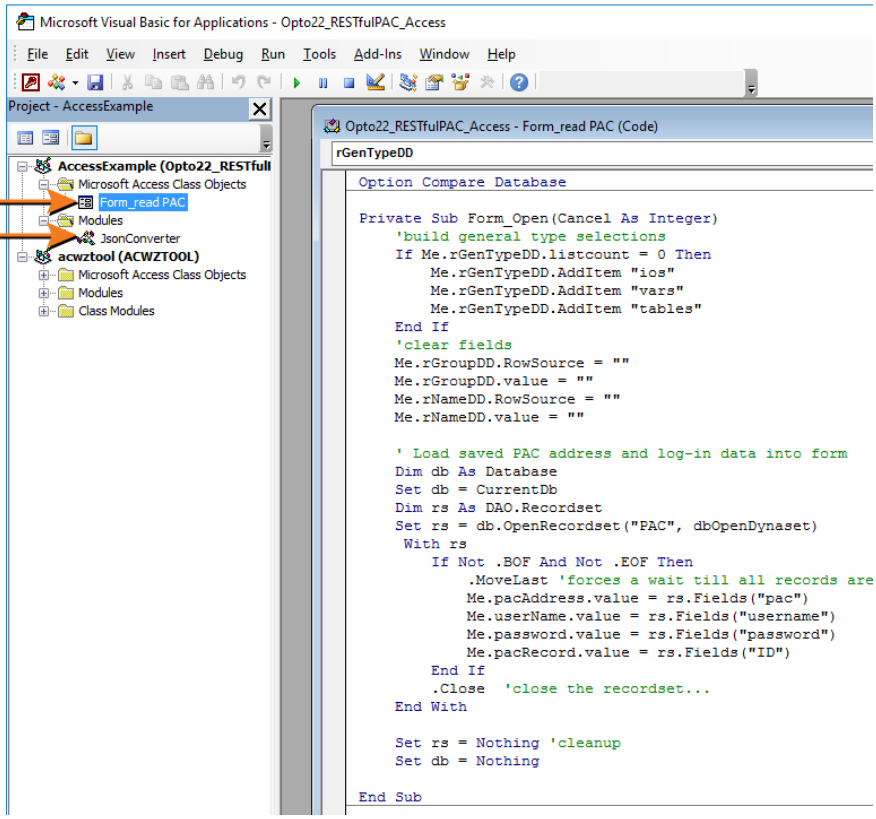
The code that requests data from the PAC and stores it in tables resides in subroutines triggered by events in the read PAC form. All the subroutines are contained in one Microsoft Access Class Object that you can open and edit, if you wish, using the VBA editor.

To see the Visual Basic code, open the Visual Basic Editor:

1. Click the Access Database Tools tab.
2. Select the Visual Basic icon.

All the custom code is contained in this object (shown).

The JsonConverter is a module written by MIT and made publicly available. It may be used according to terms of its copyright statement.



VBA references

This example requires the libraries shown below, which you can verify by choosing Tools > References in the Visual Basic editor:

- Visual Basic For Applications
- OLE Automation
- Microsoft Office 16.0 Object Library
- Microsoft Office 16.0 Access database engine Object Library
- Microsoft Internet Controls
- Microsoft WinHTTP Services version 5.1

- Microsoft XML v6.0
- Microsoft Scripting Runtime

Visual Basic subroutines

The Visual Basic subroutines are run from events in the read PAC form. All the subroutines are named for an object on the form (for example, "Form") and event (for example, "Open"). When you open the database, a macro opens the read PAC form, which in turn runs a subroutine attached to the Form_Open event.

Once a PAC address and login credentials have been entered in the form, you can request data from the PAC by selecting the type and item of data, starting with the general type, then group, and then name.

Once you select a group or name, a request for that group or name is sent. This illustrates how to collect and store a group as well as a single data item.

After you've selected a group or name, you can use the timer function to automate data collection. The timer function restricts the interval to between 5 and 30 seconds and checks that the current data selection is complete. These restrictions are a precaution against creating a fast, automated loop, which could lock up your computer.

In addition to constructing URL paths and parsing the return data, the Visual Basic code uses Access database object and SQL commands to write to and read from the tables. If you're developing a custom database, comments throughout the Visual Basic code will help you identify code you may want to use or change.