## Prelim Pages

Version Control /Approval etc

## Introduction

A number of algorithms are used in the spreadsheet for the purpose of automating a number of tasks. The major algorithms are explained mathematically for documentation and maintenance purposes.

## Target Audience

This chapter assume an intermediate VBA programming competency and experience with:

* Check sum calculations
* Nested while loops that use control data (column or row),
* Recursive algorithms which pass one or more parameters in each iteration, and
* Breadth-wise recursive methodologies.

## AGS Numbers

Real AGS numbers are issued by the Australian Public Service Commission (APSC). The algorithm has been utilised to assist in generating acceptable numbers for test purposes.

### Issuing Valid AGS Numbers

AGS numbers utilise a checksum (CS) based on the weighed sum modulus[[1]](#footnote-1) 11. The AGS number is valid when the CS value is 0. Mathematically a valid 8 digit AGS number is expressed as:

Where:

dn is the nth digit of the value.

n is the column, start with 0 at the rightmost or Least Significant Digit (LSD).

### Generating a pool of values

The check sum is the residual value needed to achieve a modulus 11 of zero, without modifying the existing 7 digits. A valid AGS number is generated by using an incrementing 7 digit number and calculating the value of d0.

The result is conditionally checked for one of three possible values:

1. One (1) will result in a calculated residual of 10, which cannot be represented in a single digit, hence discarded from the pool;
2. Zero (0) means there is no calculated residual, and;
3. Other values are subtracted from 11 meet the initial

Where:

is the calculated checksum

PCS is the partial checksum

The algorithm is implemented as a VBA class which must be instantiated before use.

### Magic Numbers

A quick way to generate a pool is to use the pattern:

abcxyabc

Where:

abc is any sequence of numbers, eg 123

xy is a series of values between 00 and 99. Valid values are:

00, 19, 27, 35, 43, 51, 78, 86.

This method is be harder to maintain a pool over time and is restricted to around 8000 useable values.

## Default Data processing

Default data in conjunction with the “force” column determines if a column should be updated or not. The routine is called “fillCellValue” which processes per Figure 1 below.



Figure

## Export Routines

Exporting from multiple sheets has two functional requirements:

* Export all sheets to QTP on one sheet
* Export unformatted or formatted to multiple sheets

To maximise flexibility, the routines use a row wise while wrapping a column wise while loop. This allows the routines to deal with column changes without the need to re-code the routines.

### Export all to QTP

The process reads from the visible sheets and copies all the pages into one page within the routines. There is an equivalent import function to restore to multiple sheets.

### Export to Test Users

The export routines maintain the page wise organisation of the source data, and

* Either keep the source (raw) format or
* Exports in a user designed format for the test users.

There is no equivalent import function for this action.

## Import Routines

Importing has two functional requirements:

* Import from QTP results to one or more sheet in excel
* Structure Builder

To maximise flexibility, the routines use a column-wise while wrapping a row-wise while loop. This allows the routines to deal with column changes without the need to re-code the routines.

While generation of a structure is not technically an import function, it has been left in that location for convenience on the ribbon. The tool is discussed in more detail in Org Tree Routines section.

## Org Tree Routines

A number of routines support this sheet feature. The form buttons initiate five (5) primary routines:

* Add tree view nodes
* Remove tree view nodes
* Save tree to XML
* Read XML to Tree
* Build Data from Tree

Each one performs their tasks with a breadth-wise recursive algorithm. Recursion was chosen due to the high re-use and process simplicity.

### Add tree view nodes

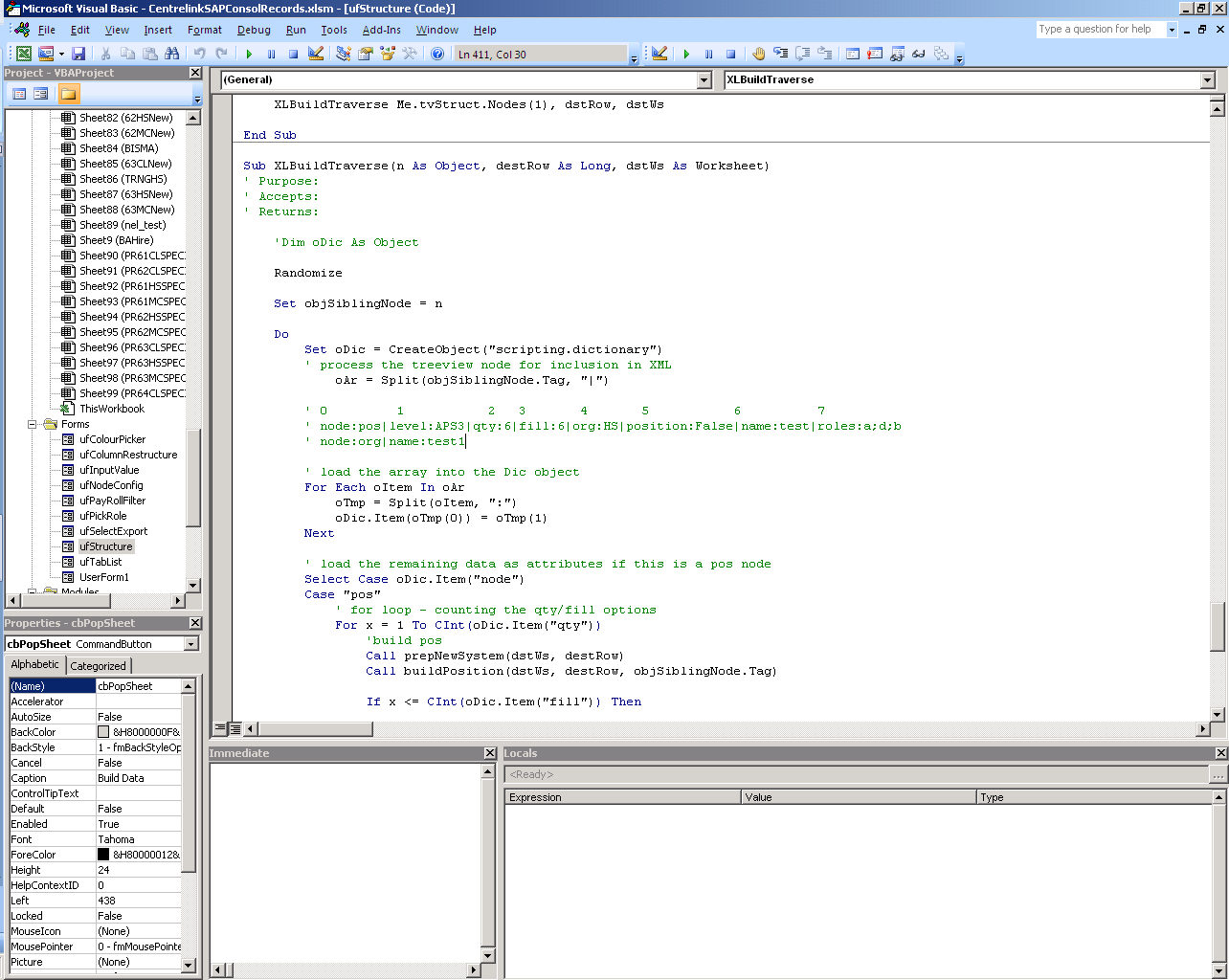
The algorithm detects if the tree is empty or not and tries to add nodes based on the users selection. It controls display of a form which guides users in the data collection and stores that information for use later in the process.

The form is responsible for converting the control values into a string handed back to the calling routine. Figure 2 below is a pictorial definition of Fields 0 to 6 and optionally 7.



Figure

An example of the form’s algorithm output is shown in Figure 3 below.



Figure

**NOTE**

The strings do no terminate with a delimiter.

These formats are used when building new data and should not be altered without consideration to the processing codes operation.

### Remove tree view nodes

The code under the Remove node button deletes the selected tree node. It should be noted that the deletion removes the selected node and all child nodes attached.

### Save tree to XML

Figure 4 below is the simplified conceptual flow chart for saving an XML representation of the visual tree to disc for later use.

This algorithm:

* Breadth-wise reads the entire tree view,
* builds an XML document in memory, and
* Saves the document using a piece of data from the user as the file name.



Figure

### Read XML to Tree

Figure 5 below is the simplified conceptual flow chart for reading a saved XML file and rebuilding the visual tree view for updating or building data.

This algorithm:

* reads a user selected XML file, and
* builds the tree view using the XML data.



Figure

### Build Data from Tree

Figure 6 below is the simplified conceptual flow chart for building new data from the Org Tree on the form.

This algorithm:

* creates a new excel sheet, using data on the custom form
* breadth-wise reads the tree and the internal stored data, and
* build the org, position and employee data from combinations of:
  + user defined default data,
  + randomly selected data and
  + users supplied data.



Figure

...oooOOOooo...

1. Modulus is the **remainder** of division by a divisor. E.g. 13 Modulus 3 = 1 (ie, 13/3 = 4 remainder 1) [↑](#footnote-ref-1)