|  |
| --- |
| Visio Automation Tool |
| V 0.10 BETA v0.1  15 March 2012  This template uses dynamic fields, title and print date. It also uses a custom property, **client** <Edit this feild to hold the Customer Name> to hold the client name. Delete this paragraph after use.  The text block below needs to have space fillers depending on the size of the logo if used. Delete this paragraph after use. |
|  |
| This document is proprietary to Name this Customer.  May not be reproduced or distributed without written permission. |

Document Revision History

This is a snapshot of an on-line document. Paper copies are valid only on the day they are printed. Refer to the author if you are in any doubt about the currency of this document.

Any substantial changes to this documented should be reflected below within the matrix.

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision Number** | **Revision Date** | **Summary Of Changes** | **Editor** |
| 0.10 BETA | 15 March 2012 | Initial Version | Ian Tivey |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Visio Automation Tool 1](#_Toc319590631)

[Introduction 4](#_Toc319590632)

[1.1 Introduction 4](#_Toc319590633)

[Using the Tool 5](#_Toc319590634)

[1.2 How the tool works 5](#_Toc319590635)

[1.2.1 Hierarchical Layout 5](#_Toc319590636)

[1.2.2 Object Dependencies 6](#_Toc319590637)

[1.2.3 Layout 6](#_Toc319590638)

[1.3 Stencils 6](#_Toc319590639)

[1.4 Excel Sheet 6](#_Toc319590640)

[1.4.1 “Nodes” Tab 7](#_Toc319590641)

[1.4.2 “Dependencies” Tab 8](#_Toc319590642)

[1.5 Running the Tool 9](#_Toc319590643)

[1.5.1 Common Errors 10](#_Toc319590644)

[Code Explanation 11](#_Toc319590645)

# Introduction

## Introduction

The goal of the Citihub Visio automation tool is to accelerate the creation of Visio diagrams depicting systems architecture and message flows, commonly created during the output of AAA and other technical engagements.

The goal of the first release is to accelerate the drawing and linkage of objects in Visio, with layout of objects being of secondary importance.

Subsequent releases will enhance the layout functionality of the tool; the vision being, that there will be several different layout functions depending on the style of diagram being created.

With this in mind, the first release has focussed on ensuring the object model is sufficient to allow for complex and varied creation of layout functions.

# Using the Tool

## How the tool works

In order to get the best results out of the tool it is worth spending a few minutes understanding how the tool works.

When gathering requirements for the tool it was clear that there were a varied set of layout requirements. Whilst the first release of the tool is not terribly sophisticated in how it lays out the objects on the page the first release of the tool has focused on providing flexibility for future enhancements to specific layouts.

The tool works on two important concepts:

* Objects are laid out hierarchically (i.e. contained within other objects, to multiple levels)
* Objects need to be connected (object dependencies)

### Hierarchical Layout

Take the following example, which shows hierarchical layout. London Data Center is the top level object. DMZ and Application Zone are child objects of “London Data Center”. The individual network components then sit within the “DMZ” and “Application Zone” objects

|  |  |  |  |
| --- | --- | --- | --- |
|  | London Data Center |  |  |
| DMZ |  |
| Server 1 |
| Server 2 |
| Ethernet |
| Application Zone |  |
| Firewall |
| Ethernet |
| App Server |
| Database |

When setting up the Excel spreadsheet you will need to think about the object hierarchies you want to draw.

### Object Dependencies

Each connection line between two objects is known as an object dependency. Any object can be connected to any other object, regardless of each object’s level in the hierarchy.

### Layout

The current layout function uses the “Containers” feature, added in Visio 2010. Every object in the hierarchy which contains other objects is turned into a Container.

Layout is an area of future improvement.

## Stencils

Section needs finishing

As part of the first release a set of Citihub approved stencils have been included

|  |  |
| --- | --- |
| **Stencil Name** | **Shape** |
| <name> | <img> |
|  |  |
|  |  |
|  |  |

## Excel Sheet

Taking care when constructing the Excel sheet will produce better results!

There are two tabs on the Excel sheet of importance. The following sections describe how to fill in each tab.

**NOTE: The VBA code uses node IDs to create hierarchy and dependency links. Textual descriptions are used for lookup formulae in the Excel spreadsheet only.**

### “Nodes” Tab

The “Nodes” tab defines the objects to be placed on the page and object hierarchy.

|  |  |
| --- | --- |
| **Field** | **Description** |
| Id | Numerically increasing identified for the node, starting at 0 for root node |
| Name | The name of the node, also the text which will appear on the node in the diagram. Some of the excel helper functions included in the spreadsheet will only work if this is a unique value. |
| Stencil | The name and/or location of the stencil file from which to take the shape for this node |
| Type | The name of the stencil shape in the stencil file |
| Parent\_node | The name of the parent node for this node. Note that all nodes, apart from the root node, must have a parent |
| Parent\_node\_id | ID of the parent node (there is a formula which looks up the parent node id based on the name of the parent node) |

To take the previous example, here is an example spreadsheet to produce this diagram.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Id** | **Name** | **Stencil** | **Type** | **Parent\_node** | **Parent\_node\_id** |
| 0 | Root |  |  |  |  |
| 1 | London Data Center | C:\visio\dc.vss | Data\_Centre | Root | 0 |
| 2 | DMZ | C:\visio\dc.vss | Zone | Root | 1 |
| 3 | Application Zone | C:\visio\dc.vss | Zone | Root | 1 |
| 4 | Server1 | Servers.vss | Server | DMZ | 2 |
| 5 | Server2 | Servers.vss | Server | DMZ | 2 |
| 6 | 192.168.2.0/24 | Network.vss | Ethernet | DMZ | 2 |
| 7 | Firewall | Network.vss | Firewall | Application Zone | 3 |
| 8 | 192.168.5.0/24 | Network.vss | Ethernet | Application Zone | 3 |
| 9 | App Server | Servers.vss | Server | Application Zone | 3 |
| 10 | Database | Servers.vss | Database | Application Zone | 3 |

Note that the top-level object, London Data Center, has the root object as its parent.

### “Dependencies” Tab

The “Dependencies” tab defines the connection points between objects.

|  |  |
| --- | --- |
| **Field** | **Description** |
| From | Name of the node to start the connection point from |
| From ID | ID of the “From” node (there is a formula included to look this up from the nodes tab) |
| To | Name of the node to connect to |
| To ID | ID of the “To” node (there is a formula included to look this up from the nodes tab) |
| Name | *Currently not in use – use default value* |
| Direction | *Currently not in use – use default value* |

To create the connections in the example the following spreadsheet could be used

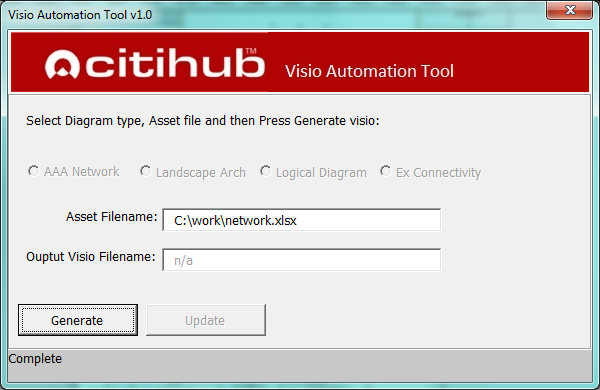
|  |  |  |  |
| --- | --- | --- | --- |
| **From** | **From ID** | **To** | **To ID** |
| Server 1 | 4 | 192.168.2.0/24 | 6 |
| Server 2 | 5 | 192.168.2.0/24 | 6 |
| 192.168.2.0/24 | 6 | Firewall | 7 |
| Firewall | 7 | 192.168.5.0/24 | 8 |
| 192.168.5.0/24 | 8 | App Server | 9 |
| 192.168.5.0/24 | 8 | Database | 10 |

## Running the Tool

To run the tool

1. Open the Visio document
2. Ensure Macros are enabled (usually the yellow bar at the top will pop up, requiring you to click “Enable”)
3. Press Alt-F11 to open the macro editor
4. Under “Forms”, double click on UserForm1
5. Select the form, by clicking on the dialog box header
6. Click the “Play” button
7. Under “Asset Filename” enter the fully qualified windows location of the excel file, including filename and extension
8. Output Visio Filename and Document Type selectors are currently in-operable
9. Click “Generate”
10. If successful you will see a “Complete” message in the bottom right

The tool uses the Visio “auto layout” function, which usually produces a fairly random layout. It usually results in the correct components being in the correct containers, but will it need some time to layout the components manually.



### Common Errors

“Unable to connect to the data” – The Excel spreadsheet could not be opened

“Invalid document identifier” – A stencil file could not be opened

# Code Explanation

*TBC*

***ShapeNode* class**

For each row in the Excel “nodes” tab the system creates a ShapeNode object in which to hold all of the associated data for that node.

The shapenode class is a set of setters and getters for the object.

***Module1***

Module 1 is the guts of the code.

*RunMe*

This is the main sub routine in the system and calls the following sub-routines in the following order:

buildNodeTree -> linkShapes -> layoutShapes

*buildNodeTree*

This routine recursively builds a tree of shapenode objects, indexed by nodeId. Each shapeNode has two sets of children:

* leftChildren: child nodes with nodeId less than the current nodeId
* rightChildren: child nodes with nodeId greater than the current nodeId

The reason the child nodes have been split out like this is to improve the performance when searching the tree for a specific nodeId.

The routine initializes the “nodes” dataset using the Visio.DataRecordset object, which imports data from the nodes tab in the Excel spreadsheet. It then iterates through the rows of the imported dataset, creating a new ShapeNode object for each row and adding it to the appropriate point in the tree using a recursive algorithm.

At the end of this subroutine all of the objects will have been dropped on the page at the same (x,y) coordinate so will be stacked on-top of each other.

*linkShapes*

This routine is responsible for drawing connectors between objects. It initializes the “dependences” dataset using the Visio.DataRecordset object, which imports data from the “dependencies” tab in the Excel spreadsheet.

For each line in the dependencies dataset it then searches the node tree for the shapenodes with nodeId=“fromId” and nodeId=”toId”, and connects the two shapes associated with those shapenode objects.

*layoutShapes*

This is the first attempt at a layout function. It is envisioned that eventually several layout functions will exist for different document-types.

This particular layout function traverses the node tree and turns any node which has child nodes into a container. As it reaches a node it adds the shape to the parent node container.