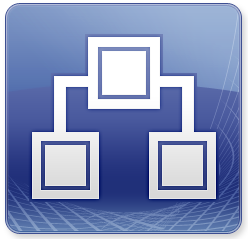
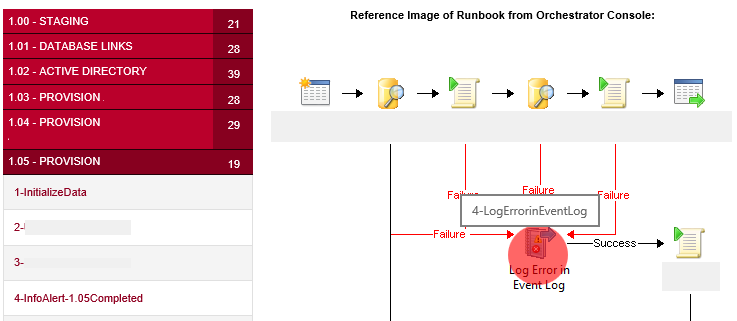
System Center Orchestrator 2012 R2  
HTML Documentation Tool v0.60 BETA

Users Guide & Reference Document





**Author:** Michael Adams – <http://willcode4foodblog.wordpress.com>  
**Last Updated**: 10-25-2014

**Contents**

[Release Notes 4](#_Toc402004515)

[What the tool does 5](#_Toc402004516)

[Why doesn’t the script do \_\_\_\_ 5](#_Toc402004517)

[Prerequisites 5](#_Toc402004518)

[Features of this script 6](#_Toc402004519)

[Generating Runbook Reference Reports 7](#_Toc402004520)

[Create the Runbook Export 7](#_Toc402004521)

[Create Your Runbook Images 9](#_Toc402004522)

[Create Your Activity Icon Images 10](#_Toc402004523)

[Update the **PARSE.OIS.Exports.config** file 10](#_Toc402004524)

[Executing the script 11](#_Toc402004525)

[Navigation: Table of Contents (Index.html) 12](#_Toc402004526)

[Navigation: Activity Pages 13](#_Toc402004527)

[Navigation: “Link” Activity Pages 14](#_Toc402004528)

[Script Components 15](#_Toc402004529)

[Component – Template Files 15](#_Toc402004530)

[Activity/Object Master Template File 15](#_Toc402004531)

[CSS Menu Template File 15](#_Toc402004532)

[General Activity Template File 15](#_Toc402004533)

[Link Activity Template File 16](#_Toc402004534)

[.NET Script Activity Report Template File 16](#_Toc402004535)

[.NET Script Published Data Template 16](#_Toc402004536)

[SQL Database Query Report Template 16](#_Toc402004537)

[Runbook Table of Contents Header Template 16](#_Toc402004538)

[Runbook Table of Contents Footer Template 16](#_Toc402004539)

[Component – Runbook Image Files 17](#_Toc402004540)

[Creating the Runbook Images 17](#_Toc402004541)

[Known Issues with Runbook Image Creation 19](#_Toc402004542)

[Component – Activity Icon Files 20](#_Toc402004543)

[Activities Folder 20](#_Toc402004544)

[Activity Icon Files Included with tool 22](#_Toc402004545)

[Misc. Folder 22](#_Toc402004546)

[Component – Configuration File 23](#_Toc402004547)

[[ General ] Section 23](#_Toc402004548)

[[ CSS ] Section 23](#_Toc402004549)

[[ Templates ] Section 24](#_Toc402004550)

[[ SQLConnectionInfo ] Section 25](#_Toc402004551)

[Performance Considerations 26](#_Toc402004552)

[Memory Usage 26](#_Toc402004553)

[CPU Usage 26](#_Toc402004554)

[Appendix A: Best Practices for Runbook Design 27](#_Toc402004555)

[Appendix B: Blog Post 1 – Example Runbook Walkthrough 29](#_Toc402004556)

[Appendix C: Blog Post 2 – Generate Documentation for Example Runbook 29](#_Toc402004557)

# Release Notes

|  |  |  |
| --- | --- | --- |
| Date | Version | Fixes |
| 10-23-2014 | 0.60 (Beta) | * Rewrote table of contents to output as **Index.html** vs **Index.html** * Fixed CSS Menu Bugs (Too many links were being added to menus) * Include Master Runbook in Export * Corrected CompanyLogo issue * Introduced **ShowNullOrEmptyProperties** Configuration Item (General Section) |
| 10-18-2014 | 0.50 (Beta) | * Fixed Issue with using folder names instead of runbook names * Corrected Left Side CSS Menu bug that appeared after initial rewrite of code * ~~Link Drop Down Menus are now not properly created (New Bug)~~ |
| 10-15-2014 | 0.40  (Beta) | * Added Drop Down for "Execution Data"  (Available Options on an Activity) * Removed "**areaLinkSource**" green mappings on the Index.html files * Removed “skeleton table” for published data for **non-script** activities |
| 10-6-2014 | 0.35  (Beta) | * Converted script globals (most) into configuration items in the **Parse.OIS.Exports.config** file * Began Users Reference Guide for tool |
| 10-1-2014 | 0.30  (Beta) | * Translated GUIDS to human readable text by using pieces from the code published for the **SMART Documentation** Tool from Microsoft. * Left SourceID, TargetID, UniqueID fields in Link Activities unconverted (*to ease troubleshooting – now you have GUIDs in one place for the source, link and target*) |
| 09-27-2014 | 0.25 (Beta) | * Fixed **Index.html** Menu issue * Corrected problems with **CSS Menu** * Created GeneralActivity function to take care of 99% of activity report generation |
| 09-3-2014 | 0.20  (Beta) | * Ready to demo * Reports Generate properly if Following Best Practices * Enhanced Logging * Images Embedding Properly |
| 08-30-2014 | 0.10  (Beta) | * Initial Build * Image mapping plausible if following a process to generate the images. Project worth continuing. |

# What the tool does

This tool is a PowerShell-based script that will assist a user in auto-generating an interactive HTML document set for each runbook and its corresponding activities. These documents can later be used as a reference when troubleshooting and a point in time capture of configuration settings you wish to validate against the current runbook settings.

This tool was originally just a **proof-of-concept** - but I think it is at the point that I need to just publish the code I have and let the community enhance it if desired.

# Why doesn’t the script do \_\_\_\_

or “I get errors running it outside of the convention used in the examples”

I am running out of time to do some of the other things I wish to do with this script (possibly build a simple front end and dynamic “viewing” utility of runbook information, etc.). My personal life and “day job” duties have been increasing in workloads exponentially over the last few months…

With that said - I believe it has evolved to a point where it will be at least of academic value/use to others – so I didn’t want this project to become another dusty archived project of mine that never saw the light of day.

Unless the conventions suggested below (Runbook layout, Design Practices designated as \**Important*\*, etc.) are followed, I cannot guarantee the results of the script execution. There should be no damage to the OIS export file during the process regardless – as the information is read into memory and then accessed from that point on in memory.

So some tweaking of conditions/flow of some code may need to be performed in order for the script to properly build a document set for runbooks that are designed in a different style.

# Prerequisites

* The tool must be run on a 64-bit machine with PowerShell – The script was written with PowerShell 4.0 on a Windows Server 2012 R2 machine, but tested on Windows 8.1
* On **pre-**Windows Server 2012 R2 operating systems, you will need to install Windows Management Framework 4.0, which also requires [.NET Framework 4.0](http://www.microsoft.com/download/details.aspx?id=30653) : <http://www.microsoft.com/en-us/download/details.aspx?id=40855>
* No Orchestrator components are needed on the machine running the tool, **but you will need network access to the Orchestrator database** (read access only).
* Images of Runbooks and activity icons (*see sections “****Runbook Image Files****” and “****Activity Icon Files****”*)
* A web browser (preferably Firefox or Chrome – *but IE11 works just as well*)
* [Trace32.exe](http://www.microsoft.com/en-us/download/details.aspx?id=9257) or [CMTrace.exe](http://www.microsoft.com/en-us/download/details.aspx?id=29265) (To view log files on the fly)

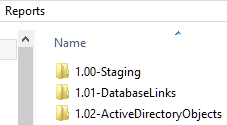
**PLEASE READ ALL OF THE DOCUMENT BEFORE RUNNING THIS TOOL**…is the central theme…I will be trying to get across in this presentation… ;)

# Features of this script

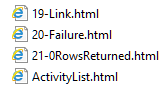
This script tries to build upon the ideas used to create the [S.M.A.R.T Documentation Tool](http://blogs.technet.com/b/privatecloud/archive/2014/05/08/updated-tool-smart-documentation-and-conversion-helper-for-your-orchestrator-runbooks.aspx), but with a little more “user friendliness” built in. I wanted to provide a method that would provide a versatile Systems Administrator or high level Operator *(see: not necessarily versed in Orchestrator design concepts)* the ability to quickly and easily troubleshoot/review/audit runbook configurations (the last known good ones – in any case) of runbooks that have been processed using this script. This has the added benefit of providing this capability with only read access to the files and a modern web browser!

This script builds out a folder structure in relation to your Orchestrator “Runbooks”. Once the folder is created (*built from the runbook name itself*), it is populated with HTML files that correspond to that particular runbook:

**Example *(below):*** Runbook folders generated in the “**Reports**” folder of the script’s root directory



**Example *(below)*:** Activity Files Generated within a report’s runbook folder



This script will take a runbook export (OIS) file, and by utilizing that information - as well as a few queries to the SCORCH database – will generate a navigable set of HTML files, that use CSS and JQUERY to provide an interactive "Map" of the runbook and the corresponding objects/activities in that runbook.

When a user clicks on the activity icon in one of the "**Index.html**" files, they will be sent to a web page created specifically for that activity. Each activity will be highlighted on an (embedded into the html file) image of the current runbook map, and will show a user the following:

* A built-in CSS drop down menu with links to each "Link" Activity connected to/from the current Activity
* Properties for each activity (*Name, ObjectType, Timeout values, etc.)*
* ~~Any~~ Most configured data from the user in a format that is quickly readable
* Format Script Code from .NET activities with CSS to allow easy reading with comments "highlighted"
* Format SQL Code in the same manner as .NET Script Activities
* Any Custom Start Parameters from an Initialize Data Activity
* Any Data sent to the data bus from a "Return Data" activity
* Any Published variables from Script Activities.
* Object GUIDs converted to human-readable text
* Link Conditions converted to human-readable "pseudo functions"

# Generating Runbook Reference Reports

\*Important\*

Because the design, preparation and configuration of your runbooks is critical to providing relevant information, I HIGHLY recommend that anyone who hasn’t read this [TechNet blog](http://blogs.technet.com/b/privatecloud/archive/2013/03/08/automation-system-center-2012-orchestrator-best-practice-series-naming-conventions.aspx) or [this document](https://scorch.codeplex.com/releases/view/119073) on Orchestrator Design Best Practices, that you stop now and read this document and follow the conventions it tries to instill.

For a quick list of tips to have “cleaner” runbook documentation generated from these scripts – see the **Appendix B:** **Example from Blog** secton at the end of this document.

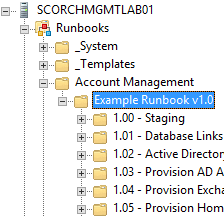
\*Important\*

This will ensure that this script provides the best value (*descriptions accurately filled in, etc*.) and with the added bonus of “*if you do it right the first time*”, then the script brings those details over with the reports it generates! You say you want reference docs for how a particular runbook operates! No problem!!! Give me 20 minutes! (Estimated time includes snipping important activity icons and all runbook images.

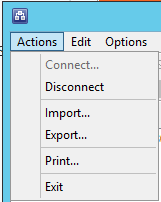
## Create the Runbook Export

The first step in the process of documenting a finished runbook is to export the runbook into an OIS\_Export file. This file can then be used to generate the documentation for that runbook.

First, open up the Runbook Designer and select the runbook folder set you wish to export. In this example, I am looking to generate reports for all of the runbooks underneath the “*Example Runbook v1.0*” folder:



**Note**: Only the runbooks in the folder *underneath* the “**Example Runbook v1.0**” folder will be documented. In order to report on the higher level runbook (sometimes called the “Master Runbook) you will need to select the folder ***above*** the “Ex**ample Runbook v1.0**” folder.

Once you have selected the root folder that contains the runbooks you wish to export, click **“Actions” > “Export”** from the **Runbook Designer** menu:  


For the **File Location,** save the file as whatever you want:

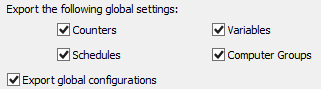


**Check** the “**Export the runbooks in sub folders**” option:  

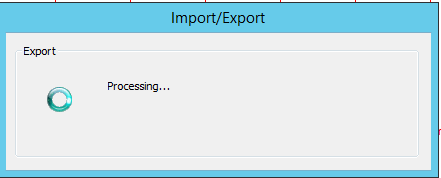
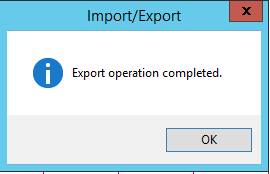

It’s your call to include a password or not – we’re not going to need the file after the report generation, although it ***could*** be retained for a copy of the exact runbook that your documentation will be representing.

When “*scrubbing*” a runbook in order to prepare it for import into a separate environment, you are guided to **un-check all** the “***Export the following global settings***” option (*and for good reason!*).

We need any pertinent information for our documentation – **so we WILL be CHECKING each of the options** (*Just remember the export will only be good when importing to the SAME environment in the future*):

  
**Example *(Above)*:** Select all options for the export so the script can read any settings it can locally

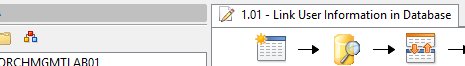
When you have all your options configured, click the **Finish** button to complete the export.

## Create Your Runbook Images

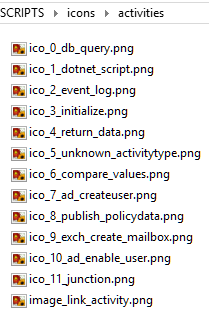
Following the steps outlined in the section “**Component – Runbook Image Files**”, create a set of runbook images and name them as according to the **FOLDER THEY RESIDE IN – NOT THE RUNBOOK NAME ITSELF**

This should be used for runbook image generation



## Create Your Activity Icon Images

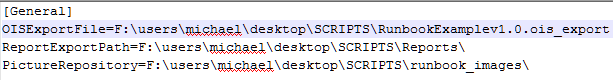
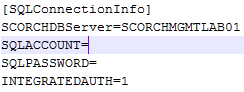
Following the steps outlined in the section “**Component – Activity Icon Files**” (*later in this document*), create a set of runbook images and place them in the **<SCRIPTDIRECTORY>\icons\activities** folder:



**Note:** The “**image\_link\_activity.png**”, and “**ico\_5\_unknown\_activitytype.png**” files need to be here or you may receive many broken images in your document set.

## Update the **PARSE.OIS.Exports.config** file

Update the configuration file accordingly. Most often these will be the only configuration items you need to set:

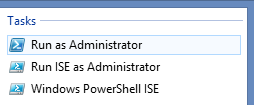
  


**Example *(Above)*:** Configurable items in the configuration file

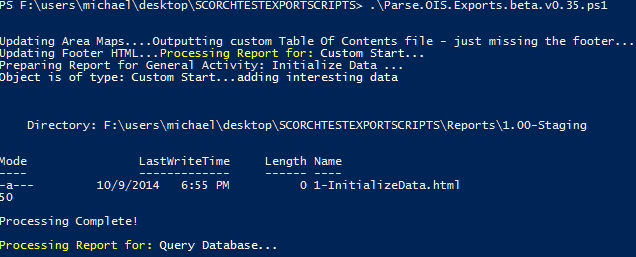
*(Note how we’re being good boys and girls by using* ***Integrated Authentication****)*

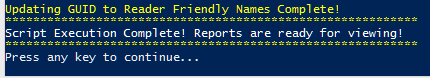
## Executing the script

Once you’ve got all your settings correctly configured, you can fire up a **PowerShell Session** or even execute the script through the **PowerShell ISE** if you desire.

****

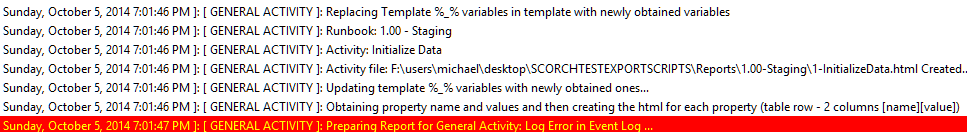
Just call the script from the console *(or click the run button in ISE)*and **wait till the script closes the PowerShell window reports that processing is complete:**



**This process can take a little time, but I generated 170 pages of runbook documentation within 5 minutes – so it’s worth the wait…just wait for the script to finish:**

If you want a little more info on what’s happening during the script execution, you could go to the **“log”** folder and open up the log file with **CMTrace.exe or Trace32** (listed in **Prerequisites** section of this document)

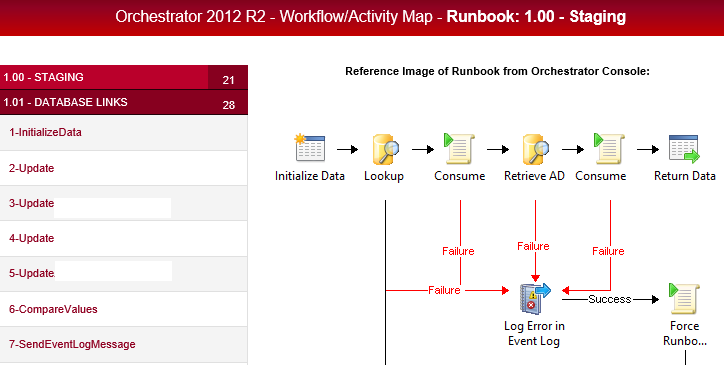


 **Example *(Above)*:** Sample output from a log file that is a little more verbose than the standard output of the PowerShell Console session

Notice in the example above how it **looks like we have an error**.   
***THIS IS NOT NECESSARILY THE CASE!*** (see next page)

You need to be aware that this utility will automatically highlight in red any line containing “Error”, “Failure”, etc. So if your activity is named something similar (*as in the last line from the previous example – with an activity called “Log* ***Error*** *in Event Log”*), you just need to know everything’s ok and just wait for the script to **finish** before looking through anything (**don’t lock ANY files the script is using until it has completed in its entirety**)

# Navigation: Table of Contents (Index.html)

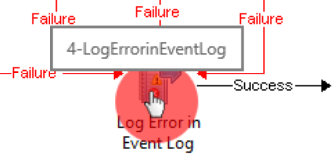


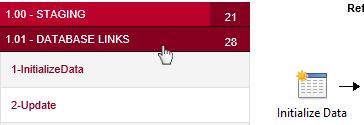
The **Index.html** file inside each generated runbook folder is the “**Table of Contents**” for that particular set of html reports.

A Runbook “**Table of Contents**” contains a navigable runbook (*click the activity to jump to its report*) and also provides a CSS “Drop Down” menu on the left hand side, where you can jump to OTHER runbooks in this “document set”.

The **Index.html** files contain the following features:

* Clickable runbook maps that let you jump straight to that activities report

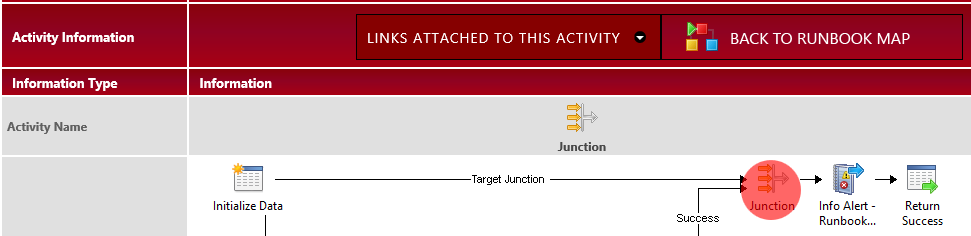
  
**Example *(Above)*:** A Runbook “Table of Contents” contains a navigable runbook   
(click the activity to jump to its report)

* Left-side **CSS Based menu** that allows you to jump between runbooks in the document set that was generated:  
    
  **Example *(Above)*:** CSS “Drop Down” menu on the left hand side

# Navigation: Activity Pages

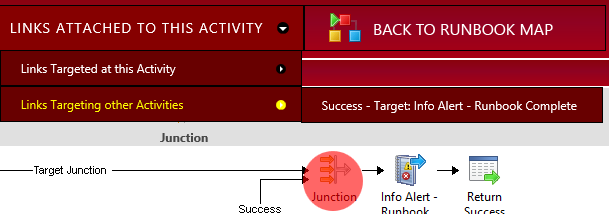
Notice how in the image below – the “Junction” activity is highlighted with a red circle. This allows the user to know which activity they are currently looking at in respect to the overall workflow/runbook:

**Example *(below)*:** Activity Page (*Junction Activity*)



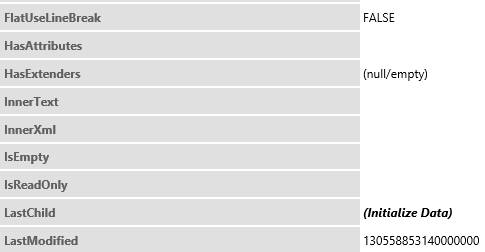
In order to see which activities are connected to this activity via links, you just use the CSS-enabled menu above the runbook image to see what activities you can “jump” to:

**Example *(below)*:** Links attached to Activity (*Junction Activity*)



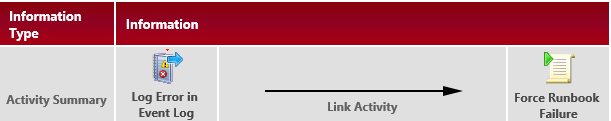
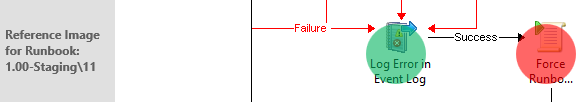
Scrolling down the page you can get a list of all the properties that were found during the processing of this particular activity:

**Example *(below)*:** Properties of an Activity (*Junction Activity*)



# Navigation: “Link” Activity Pages

The Link Activity pages are basically modified versions of a General Activity with the following differences:

* Instead of the one activity icon there are three icons representing the link relationship:   
    
  **Example *(Above)*:** The summary of a link activity shows the source, the “Link”, and the target.
* The source of the link is highlighted in Green *(think “Start”)* and the target is highlighted in Red *(think “Stop”)* on the reference runbook map on the Link Activity Page. These highlighted activities are also clickable, so you can jump to that activity’s report page:  
    
  **Example *(Above)*:** The green source of the current link, and the red target of the current link
* The Triggers *(Conditions)* of the link are presented in a format of “pseudo-code” that can be understood by anyone with an understanding of basic programming/scripting principles   
  (*the code for this piece was blatantly copied from the* [*S.M.A.R.T Documentation Tool’s*](http://blogs.technet.com/b/privatecloud/archive/2014/05/08/updated-tool-smart-documentation-and-conversion-helper-for-your-orchestrator-runbooks.aspx) *source code):*

The trigger below basically says: “If the Activity ‘**Log Error in EventLog’** returns “**success**”  
  
**Example *(Above)*:** How the script presents “Link Conditions” of a runbook

* The source, link itself, and target ID’s are not translated to “human-speak” for Link Activity reports. I decided that - by doing this - you now have 3 GUIDs you can reference when troubleshooting Orchestrator logs. Since the names are easily found by looking at the image references – there’s no need for me to translate those GUIDs.

# Script Components

The next few pages will attempt to illustrate and explain the various components/files/data that are utilized by the script. The folder structure and even the naming conventions of some files (i.e. icons) can cause unexpected results if the conventions laid out in this document are not followed.

## Component – Template Files



This script relies on multiple \*.template files. These files are chunks of HTML with unique “Placeholder” Variables (in the format of ‘*%\_VARIABLE\_NAME\_%*’) that we replace from information gathered during the script execution. The templates are important because different HTML “chunks” need to be updated at different times during report generation. It is only during the last steps of creating the activity report that all the “chunks” are combined into a final HTML report.

The following is a list of the template files and their purpose:

### Activity/Object Master Template File

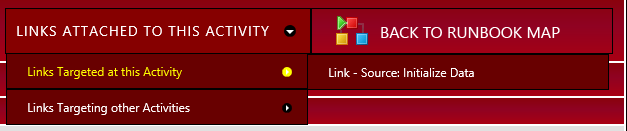
File Name: SCORCH.Runbook.DetailsReport.template

Function: This code is used as the “master page” that embodies the whole HTML report for an activity. This file includes the *html, body, head*, and *style* tags for the resulting activity HTML Report that will be produced.

### CSS Menu Template File

File Name: SCORCH.Runbooks.CSS.template

Function: This code is used as a framework to create a dynamic menu of the “Links” associated with an activity. This code is specially formatted (probably my own formatting issues) – but I couldn’t get the code to properly show in multiple browsers without removing all line breaks from the file.

Example: CSS Menu that lets users see further information regarding links attached to activity  


### General Activity Template File

File Name: SCORCH.Runbooks.GeneralActivity.template

Function: This code creates the main table structure used for all activity types that do not have special templates of their own. As of this writing those special templates are for: .NET Scripts, Database Query, and Link Activities.

### Link Activity Template File

File Name: SCORCH.Runbooks.Link.Report.template

Function: This is a modified version of the General Activity template. It differs in that in the row that identifies the activity by icon and name, there are now 3 icons which represent the link source, the link itself, and the link target activity/object. Also the link activities do not have the CSS Menu for links that all other activity reports have.

### .NET Script Activity Report Template File

File Name: SCORCH.Runbooks.PoSH.Report.template

Function: This is a modified version of the General Activity template. It differs in that in the beginning I only specified the properties I wanted. This could probably be deprecated but there were a lot of non-interesting properties (I felt at the time) that I didn’t have a solution for presenting “Cleanly”

### .NET Script Published Data Template

File Name: SCORCH.Runbooks.PoSH.PublishedData.template

Function: This code block is a formatted table that presents the data (if any) that is published to the data bus after successful completion of a .NET Script activity.

### SQL Database Query Report Template

File Name: SCORCH.Runbooks.DBQuery.Report.template

Function: This is a modified version of the General Activity template. It differs in that in the beginning I only specified the properties I wanted. This could probably be deprecated but there were a lot of non-interesting properties (I felt at the time) that I didn’t have a solution for presenting “Cleanly”

### Runbook Table of Contents Header Template

File Name: SCORCH.Runbooks.TableOfContents-Beginning.template

Function: This is the beginning of the HTML code for an “Index.html” file that each script-generated runbook folder contains. This file is an interactive map of the runbook and can be used to “jump” either directly to an activity with that runbook, or to another activity in one of the other runbooks that were contained within the whole runbook “workflow”.

It also contains the CSS you can modify to change the color themes, images, etc.

### Runbook Table of Contents Footer Template

File Name: SCORCH.Runbooks.TableOfContents-Footer.template

Function: Just a simple set of HTML tags that close out the HTML file itself. Could be modified to present other information underneath the maps – (Runbook Timers, counters, etc.) if desired.

## Component – Runbook Image Files

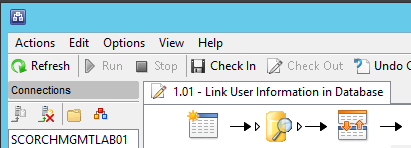


This is the location the script uses to take an image of the runbook and creates a base64 encoded string of it that gets embedded into all the corresponding activities/table of contents files that will be generated for that runbook.

### Creating the Runbook Images

I discovered that by using this method EVERY TIME I wanted to get an image of a runbook – that I could generate a block of code that could use that image as a basic “Image Map”, and allow me to provide mouse over interaction with the map via some snazzy open source JQuery code.

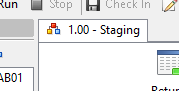
First, open up the Runbook Designer application and open up the runbook you wish to take a “snapshot” of:



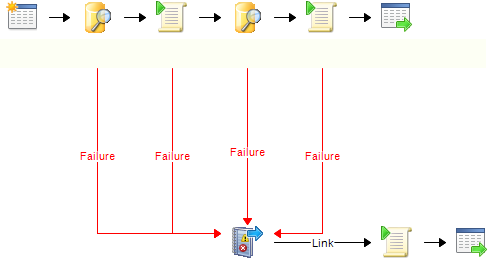
Now, open up the “Snip” tool (read more [here](http://windows.microsoft.com/en-us/windows/use-snipping-tool-capture-screen-shots#1TC=windows-8)):



Now, make the start of the “snip” at the upper left hand corner of your Designer panel (without including the borders if you can) and drag from the upper left corner (called out below) to the lower right (adjusting as necessary) until you have a snip of the runbook workflow:



Start here, and drag towards the marked arrow until you have covered the whole runbook



**Example *(above)*:** Runbook image for the script   
*(names removed and border added in MSWord – there is no border on the image itself)*

\*Important\*

NOTE:

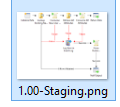
This was only tested on smaller runbooks – ***as according to best practice***. So the larger the image - the more unpredictable the results might be.

Once you have the image, save it to the “**runbook\_images**” folder with the naming format that follows the following rule (*shown in pseudo code format*):

<NewRunbookImageName> = <RunbookName>.Replace(<Space Char>, NULL) + “.png”

Say what? Basically the script will be looking for the image name that is basically the original runbook’s name with all spaces removed and with a “.png” format appended to the end. (If you do not want to use the PNG format for whatever reason you can change it within the script’s code. Just search for “.png”

In the case below, if my runbook name was **1.00 – Staging**, the filename I would use for this runbook image is “**1.00-Staging.png**”.

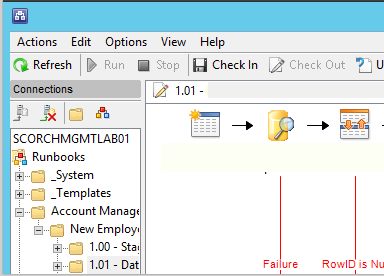


Rinse and repeat for each of the runbooks you will be generating documentation for.

### Known Issues with Runbook Image Creation

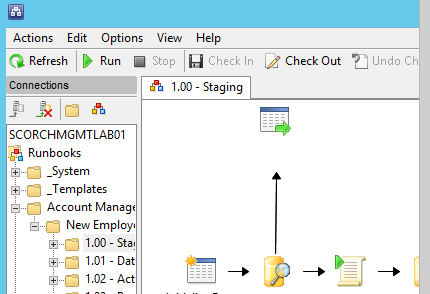
If you do not start your first activity at the upper left corner, this script will not be able to properly generate some of the x, y coordinates and your resulting image maps will not line up properly. This is something I wished to troubleshoot further but never had the time. So basically:

I know that runbooks like this will work fine:



This activity **IS** at the normal upper left “starting” corner, so **the script should work just fine!**

And I also know that runbooks like this might have issues:



This activity is not at the normal upper left “starting” corner, so you could experience issues generating documentation for this particular runbook

That’s all the known issues I currently have noted.

## Component – Activity Icon Files



The icons folder contains two sub-folders: “**activities**” and “**misc.”:**



### Activities Folder

This folder contains icons that will be used for individual activities. You create these by using the “snip” tool to take an *(approx.)* 30x30 snip of the image.

The same concept applies when creating activity images, except instead of snipping the whole runbook, you are just snipping the icon itself. I.e.:

 (Example snip of a standard “database query” activity)

Note: I used [Paint.NET](http://www.getpaint.net/images/dotPDN_DlNow.png) to remove the white and save the snip as a transparent .png file

\*Important\*

The naming standard you use is very important.

The format for naming the icons is this:

ico\_*<IndexNum>*\_*<Descriptive\_Text>*.png

The following table tries to explain the purpose for each piece of the file name

|  |  |
| --- | --- |
| Naming Standard for Activity/Object Icon Files | |
| ico\_ | Standard beginning string of icon filename |
| <IndexNum> | Index of the item in the icon image array\* |
| \_ | Underscore character (shift+dash) |
| <Descriptive Text> | A short description (can be any allowable file characters) of the activity.  (E.g.: “db\_query” or “dotnet\_script”) |
| .png | Typical png image file extension |

The resulting filename should be similar to the one below   
(*A standard SQL Query Database Activity – the first icon I made*):



\* This array is pre-defined in the script inside the “Get-IconForActivity” function. To determine which index number you should use, go to the script and find the following piece of code inside that function (next page):

*(cont. from previous page)*

switch ($object.ObjectTypeName.’#text’)

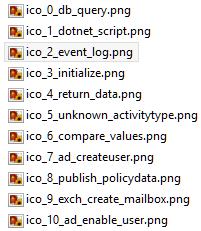
Under this switch statement is where you define which activities you have icons for.

You can add icons as necessary by using the steps mentioned above to create the icon files. Then you place it in the “icons/activities” directory, then add it to the script’s data like so:

'<Activity’s Object Type Name>' { $iconBase64 = Convert-ToBase64Pic -path "$($icons[IndexNum].FullName)"; break; }

In order to determine which index number you should be using, the easiest way would be to first inspect the path containing all the icon folders and look for the highest number. Then add 1 to that and you have your new icon’s index number!

Example, in order to add an activity icon for the Junction Activity, first I look at what is already in the folder, and find the highest valued index currently there:



According to this list, the highest value is 10. So all I have to do is add 1 to that (11) and now I have my index number. So in staying with the example for the Junction Activity, the following is true:

* The file name I need to use is “**ico\_11\_junction.png**”
* The code I add to the function Get-IconForActivity is as shown below   
  *(All one line in the script) :*

'Junction' { $iconBase64 = Convert-ToBase64Pic -path "$($icons[11].FullName)"; break; }

That’s it! Now any runbook that has a Junction Activity will use your new icon instead of the generic version (*which is used by default if you do not have a specialized icon identified for any activity*)

*🡨(Example of generic icon for unknown activity) F:\Users\Michael\Desktop\SCORCHTESTEXPORTSCRIPTS\icons\activities\ico_11_junction.png🡨 (Example of Junction Activity icon)*

HINT: The generic icon name is: ico\_5\_unknown\_activitytype.png. Just replace that file with any icon you wish to use when a current activity has no known associated icon in the script.

### Activity Icon Files Included with tool

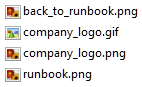
For your reference (*so you don’t think I’m leaving you high and dry*) – I have a “starter set” already going. Here’s a quick breakdown of Activity Icons already included with the script:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Icon | ObjectTypeName | Index # | Activity Icon File Name | Integration Pack |
| F:\Users\Michael\Desktop\SCORCHTESTEXPORTSCRIPTS\icons\activities\ico_0_db_query.png | **Query Database** | 0 | ico\_0\_db\_query.png | Utilities |
| F:\Users\Michael\Desktop\SCORCHTESTEXPORTSCRIPTS\icons\activities\ico_1_dotnet_script.png | **Run .Net Script** | 1 | ico\_1\_dotnet\_script.png | System |
| F:\Users\Michael\Desktop\SCORCHTESTEXPORTSCRIPTS\icons\activities\ico_2_event_log.png | **Send Event Log Message** | 2 | ico\_2\_event\_log.png | Notification |
| F:\Users\Michael\Desktop\SCORCHTESTEXPORTSCRIPTS\icons\activities\ico_3_initialize.png | **Custom Start** | 3 | ico\_3\_initialize.png | Runbook Control |
| F:\Users\Michael\Desktop\SCORCHTESTEXPORTSCRIPTS\icons\activities\ico_4_return_data.png | **Publish Policy Data** | 4 | ico\_4\_return\_data.png | Runbook Control |
| F:\Users\Michael\Desktop\SCORCHTESTEXPORTSCRIPTS\icons\activities\ico_5_unknown_activitytype.png | **Default (Unknown) Activity** | 5 | Ico\_5\_unknown\_activitytype.png | All Activities you don’t have custom icons for will use this as the default |
| F:\Users\Michael\Desktop\SCORCHTESTEXPORTSCRIPTS\icons\activities\ico_6_compare_values.png | **Compare Values** | 6 | ico\_6\_compare\_values.png | Utilities |
| F:\Users\Michael\Desktop\SCORCHTESTEXPORTSCRIPTS\icons\activities\ico_7_ad_createuser.png | **Create User** | 7 | ico\_7\_ad\_createuser.png | Active Directory Integration Pack |
| ~~F:\Users\Michael\Desktop\SCORCHTESTEXPORTSCRIPTS\icons\activities\ico_8_create_folder.png~~ | **Create Folder** | 8 | ico\_8\_create\_folder.png | File Management |
| F:\Users\Michael\Desktop\SCORCHTESTEXPORTSCRIPTS\icons\activities\ico_9_exch_create_mailbox.png | **Enable Mailbox** | 9 | ico\_9\_exch\_create\_mailbox.png | Exchange Admin Integration Pack |
| F:\Users\Michael\Desktop\SCORCHTESTEXPORTSCRIPTS\icons\activities\ico_10_ad_enable_user.png | **Enable User** | 10 | ico\_10\_ad\_enable\_user.png | Active Directory Integration Pack |
| F:\Users\Michael\Desktop\SCORCHTESTEXPORTSCRIPTS\icons\activities\ico_11_junction.png | **Junction** | 11 | ico\_11\_junction.png | Runbook Control |
| F:\Users\Michael\Desktop\SCORCHTESTEXPORTSCRIPTS\icons\activities\ico_12_get_file_status.png | **Get File Status** | 12 | Ico\_12\_get\_file\_status.png | File Management |
| F:\Users\Michael\Desktop\SCORCHTESTEXPORTSCRIPTS\icons\activities\ico_13_append_line.png | **Append Line** | 13 | Ico\_13\_append\_line.png | Text File Management |
|  | **Invoke Runbook** | 14 | Ico\_14\_invoke\_runbook.png | Runbook Control |
| 🡪 | **Link** | NA | image\_link\_activity.png | Used to represent the actual link between two objects |

### 

### Misc. Folder

This folder contains icons that will be used for stylizing the resulting report HTML files:



The “**company\_logo**” file can be changed to your own customized logo in the upper left of each report file that gets generated:



The other two “runbook” images are used for the CSS Menu links on regular Activity reports and other misc. HTML stylization.

## Component – Configuration File

This configuration file is formatted INI style – yes- a bit old school, but anyone is free to convert it to XML (*or app.config*) if desired. This file is by default located in the “Script’s Root” folder

### [ General ] Section

The **[ General ]** section contains a few global variables related to file paths the script will be using during execution:

|  |  |  |
| --- | --- | --- |
| Item Name | Value | Description |
| OISExportFile | <Path to OIS\_EXPORT File> | This is the full path to the OIS\_Export file you wish to use during script execution |
| ReportExportPath | <ScriptDirectory>\Reports | This will be the folder where all your runbook folders are created (with all activity HTML files inside the runbook folder) |
| PictureRepository | <ScriptDirectory>\runbook\_images | This folder will hold all the runbook images you created (*see previous sections*) prior to executing the script |
| CompanyLogo | <ScriptDirectory>\icons\misc | File path to the image you want to use in the upper left hand corner of the reports. |
| ShowNullorEmptyProperties | True (Default) or False | Tells the script whether to include property rows in the reports that have empty or null values. Can help reduce the information flood you get sometimes with the defaults. |

### [ CSS ] Section

The **[ CSS ]** section contains the path to the style sheet we will be embedding in the files *(so the files can be easily transported without external dependencies)*.   
**Note**: This style sheet file is by default located in the “Templates” folder

|  |  |
| --- | --- |
| Item Name | Value |
| StyleSheetPath | …\style.css |

### [ Templates ] Section

The **[ Templates ]** section contains the values of all templates and source code (JavaScript, etc.) that are used during the creation of the HTML Reports for all activities. *See the previous section – “Template Files” (page 4)* – of this document for further information about each file.

**Note**: All files are stored in <SCRIPTDIRECTORY>\Templates

|  |  |
| --- | --- |
| Item Name | Value |
| templateReport | …\SCORCH.Runbook.DetailsReport.template |
| PoshInfoTemplate | …\SCORCH.Runbooks.PoSH.Report.template |
| PoshPublishedDataTemplate | …\SCORCH.Runbooks.PoSH.PublishedData.template |
| LinkActivityTemplate | …\SCORCH.Runbooks.Link.Report.template |
| SQLInfoTemplate | …\SCORCH.Runbooks.SQLDBQuery.template |
| GeneralActivityTemplate | …\SCORCH.GeneralActivity.Report.template |
| TableOfContentsHeader | …\SCORCH.Runbooks.TableOfContents-Beginning.template |
| TableOfContentsFooter | …\SCORCH.Runbooks.TableOfContents-Footer.template |
| CSSMenuTemplate | …\SCORCH.Runbooks.CSSMenu.template |
| JQueryMenuHighlightScript | …\jquery.maphilight.js |

### [ SQLConnectionInfo ] Section

The **[ SQLConnectionInfo ]** section contains the values used to build a database connection string.

Note: Settings in **<ScriptDirectory>\Parse.OIS.Exports.config**

|  |  |
| --- | --- |
| Item Name | Value |
| SCORCHDBServer | Server name housing Orchestrator Database |
| DBIPAddress | IP address of the server (*yes it’s redundant - I don’t care*) |
| DBName | Name of the Database (*‘Orchestrator’ by default*) |
| Port | Port used by SQL (*1433 by default*) |
| SQLACCOUNT | Use only if you are not allowed to use integrated auth  (SHOULD NOT BE USED – UNSECURE STORING OF USERID)\* |
| SQLPASSWORD | Use only if you are not allowed to use integrated auth  (SHOULD NOT BE USED – UNSECURE STORING OF PW)\* |
| INTEGRATEDAUTH | 1 (*Default – True*). Set to 0 to use SQL Authentication (Needs SQLACCOUNT and SQLPASSWORD fields filled in if 0) |

*\*The option for SQL Auth is only for emergencies or temp/troubleshooting. You should be using integrated authentication for the SQL Connection.*

# Performance Considerations

I just put this section here in case anyone was interested on the “impact” my script had on my system during execution.

## Memory Usage

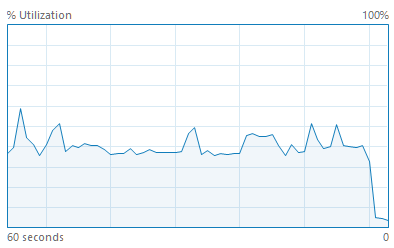
Right before executing the main script, the pre-loading of information makes the PowerShell process on my box use up approximately **26.5 MB**:  
  
During the last leg of the execution of this example runbook. Memory usage peaked at around **300+ MB** during each processing of a runbook but was steadily staying around **250 MB**:

  
(*Ram usage is the more yellow-ish color*)

## CPU Usage

CPU Usage was negligible:

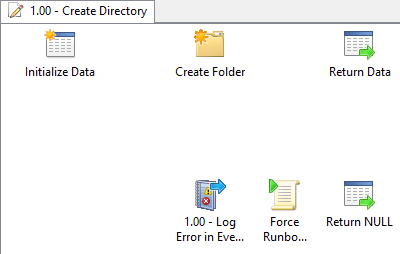
Peak Usage with an Intel Core i5-2500K @ 3.3 Ghz was **50%,** hovering mostly around **40%:**



# Appendix A: Best Practices for Runbook Design

The following is a quick list of items that you should abide by when designing runbooks. By following these guidelines you can ensure the documentation generated by this script will be its most effective for the users reading the reports.

* **Always enter description information into your runbooks** as you design them. They will carry over when the script generates the documents and provide anyone trying to understand what is happening with some more human-understandable information in there…
* **Number your runbooks in a structure** like ”**1.00 –** Runbook Start, **1.01 –** Runbook Process, **1.02 –** Runbook Complete”. Using a number prefix helps people understand the workflow at a high level by just glancing at it – rather than trying to figure out the path the linked runbooks take on their own…
* **When creating runbooks, layout your action objects first, then go back and create the links**

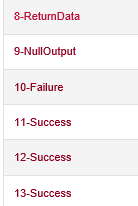
**Example** (*Below*): First add the **activities**, and lay them out without the links  


No links created until all activities were laid out

The reason for this is that the script will process the links in the order in which they were created. This doesn’t change anything about how the scripts resulting reports ***look*** to the end user, but will impact the order in which the HTML activity reports are generated.

The biggest impact is to the **Table of Contents** (*Index.html*) file for each runbook. The order of the sub-menu links (activities) under each runbook will be displayed in the order in which they were created during the design of the runbook:

**Example** (*below*)**:** Runbook Activities on the **Table of Contents** pages.



Note how the end of the activities (9-*NullOutput*) is where the links begin on this sub-menu. This helps you easily find the activities (*the idea being you want to look at an activity and can then browse to the links via the drop down menus*)

You can make sure the important items (the activities themselves) are listed at the top by designing your runbooks in a way that has you **laying out all the activities first** before going back and creating the link objects.

* **Organize your runbooks into a subfolder structure that follows the following basic guidelines:**
  + The root runbook folder contains one runbook – the **Master Runbook**
  + The subfolders are laid out in numbered order of operation (*1.00, 1.01, etc.*)
  + The subfolders contain one (or more) runbooks directly associated with the **Master Runbook**

The next appendix (*Appendix B*) is just a straight copy of my blog post titled “**System Center Orchestrator 2012 R2 -** **SCOrch Runbook Documentation Tool v1.0 Release – Post 1**”…so you can see a walkthrough of a simple Runbook Design and then documenting it with this tool.

# Appendix B: Blog Post 1 – Example Runbook Walkthrough

See the additional document of the same title (above) included with the zip file you downloaded

# Appendix C: Blog Post 2 – Generate Documentation for Example Runbook

See the additional document of the same title (above) included with the zip file you downloaded