The 2013 Scripting Games

Competitor's Guide
Welcome............................................................................................................. 3
The Tracks............................................................................................................ 4
Scoring and Winning ............................................................................................. 5
Prizes .................................................................................................................... 6
Guidelines ............................................................................................................ 8
What Not to Over-Obsess About ......................................................................... 10
Try Not to Miss the Whole Point of the Games..................................................... 11
Why You Should Vote on Entries, Even if You Think You’re Not Qualified .......... 12
Welcome
Welcome to the 2013 Scripting Games, now managed by PowerShell.org! This Guide is designed to help you understand how the games work, and what you can expect.

First up, you'll need to register for the Games by logging into [http://TheScriptingGames.com](http://TheScriptingGames.com) - note that there is no “www” on this URL. When you register, you'll be asked to select a track. This is a one-time decision, and you may choose from:

- **NONE**: You don’t want to compete, but do want to grade other folks’ work by voting
- **BEGINNER**: You’ll be able to vote on all events, but will only be able to submit one entry per event in the Beginner track.
- **ADVANCED**: You’ll be able to vote on all events, but will only be able to submit one entry per event in the Advanced track.

Events are open only for a limited period of time. Each event goes through four phases. Dates refer to midnight GMT of the indicated day (meaning, “Feb 2nd” would be “Feb 2nd at midnight GMT”). The four phases are:

- **PENDING**: The event is not yet open.
- **OPEN**: You can download the event scenario (as a PDF for offline use) and submit entries. You usually get about 5 days to do this.
- **REVIEW**: No new entries are accepted, but everyone can vote on all events.
- **PUBLIC**: No entries or votes are accepted, but anyone, logged in or not, can see the entries.

**Caution**
We ask that you not include ANY PERSONALLY IDENTIFIABLE INFORMATION in your entries. This includes your name, e-mail address, or other contact information.

**Under New Management**
Remember, the Scripting Games are now managed by PowerShell.org, Inc., a community-owned corporation that runs the PowerShell.org Web site. While the Games are still supported by Microsoft and The Scripting Guys, we’re an independent organization and our actions and words do not represent Microsoft.
The Tracks
The Beginner track consists of events where the answer is **usually** a one-liner, or at most a couple of lines of code. We do not usually expect to see error handling or error suppression, extensive use of variables, and so on. We recognize that entries in the Beginner track may sometimes produce errors (like if the command can’t connect to a computer), and that’s fine. Judges will typically be less impressed with overcomplicated solutions, so *keep it simple.*

The Advanced track consists of events where the answer is **usually** a parameterized advanced function. If you don’t know what an advanced function is, the Advanced track is not for you. We expect to see more attention to detail, and more use of built-in PowerShell features.
**Scoring and Winning**

There are several ways to win.

- Every time you vote on someone's entry (giving it a score of 1 to 5, with 1 as “bad” and 5 as “good,” whatever those terms mean to you personally), you earn one Pointlet. Each Pointlet serves as a prize raffle ticket.

- You can win by being the Crowd Favorite! That simply means more people have given you high-scoring votes as part of your CrowdScore. These aren't professional judges, but their opinion still matters!

- Our professional judging panel will select their Best And Worst list for each event, and will blog about what they liked and didn't like. If you're in the Best list for one or more judges, your entry will be reviewed by our Mighty Panel of Celebrity Judges, who will award First, Second, and Third place.

- We'll recognize the winners in each event and track, as well as the overall winners for each track.

What's it take to impress the public and earn a high CrowdScore? We've no idea. It's the *public*. Be creative and do the right thing.

What's it take to wind up on a judge's Best list? Well... have a creative approach to the problem you're given, and consider some of the Guidelines in the next section of this Guide.

And this is an important note: "Win" does not mean “prize.” Not every recognized winner will receive a tangible prize (although we're gonna try). Every winner will have the right to use a badge on their PowerShell People (http://powershell.org/people) profile, and we'll announce those badges after the Games complete. (Oh, you don't have a profile? Well, if you want to compete in the Games, there's no better rehearsal than to write the script needed to set up your People profile!)

We'd like to offer thanks in advance to our Presenting Sponsors, who are providing the majority of the prizes.
Prizes
In all cases, unless noted otherwise, we will award the same prize for the Beginner and Advanced tracks separately.

Event Prizes
These 1st prizes are awarded by our panel of celebrity judges. These judges will review the events that received the top community vote scores, but will use their own discretion in awarding prizes. There is no fixed criteria for these prizes.

Overall Winners Across All Events
1st prize: Complimentary pass (admission only; no expenses are covered) to TechEd NA 2013, TechEd Europe 2013, or TechEd NA 2014 (your choice).
2nd prize: SAPIEN Software Suite 2012 ($699 value) provided by SAPIEN Technologies
3rd prize: Five (5) ebooks (average value $200) provided by Manning

Event 6
1st prize: PrimalScript 2012 ($349 value) provided by SAPIEN Technologies
3rd prize: eBook (average value $40) provided by Manning

Event 5
1st prize: PowerShell Studio 2012 ($349 value) provided by SAPIEN Technologies
3rd prize: eBook (average value $40) provided by Manning

Event 4
3rd prize: eBook (average value $40) provided by Manning

Event 3
3rd prize: eBook (average value $40) provided by Manning

Event 2
3rd prize: eBook (average value $40) provided by Manning

Event 1
3rd prize: eBook (average value $40) provided by Manning

Crowd Favorite Prizes
These prizes are awarded to the events with the top community vote score. We will award one prize for each event, in each track.

3rd Place (all events): Ebook (average value $40) provided by Manning
Prizes for Community Voting

The top two community voters will receive a complimentary pass (admission only; no expenses are covered) to the PowerShell Summit North America 2014. “Top voters” will be identified both by the quantity of votes (in either track) and by the quality (consistency, fairness) of their votes.

In addition, the following prizes will be raffled off, with each vote cast acting as a raffle ticket:

- Four (4) $50.00 gift certificates to the SAPIEN Technologies online store, provided by SAPIEN Technologies
- Twenty (20) ebooks (average value $40) provided by Manning
Guidelines

Now, these are just ideas for you to consider. They’re not rules, and our judges won’t universally agree on every one. So don’t think of this as a magic checklist that will make you win. You don’t even need to force yourself to do all of these things – but keep ‘em in mind.

• We love to see error handling, especially in the Advanced events. But we don’t like error suppression. That is, if an error occurs, we should either see the error, or you should be doing something about it like logging it or displaying an alternate message. The exception: it’s okay to suppress an error that means “everything is going fine.” For example, you try to delete a file that doesn’t exist, and get an error. Well, that’s fine. At the end of the day, the file isn’t there, and that’s what you wanted. If you do choose to handle an error, make sure you’re doing so intelligently. Don’t wrap your entire code in a big Try{} block. Only wrap the commands that you anticipate having an error, and Catch{} (and handle) that error.

• We hate seeing people do more work than is necessary. For example, you shouldn’t ever prompt the user for a missing parameter. Use PowerShell’s Parameter() decorator to mark the parameter as mandatory.

• We love self-documenting scripts. That doesn’t just mean comments – although those are pretty awesome, too, especially comment-based help. Using full command names, full parameter names (and no positional parameters), stuff like that makes a script more self-documenting. Other examples are less obvious. Here’s one: if you’re accepting input arguments, do so via declared, named parameters, not by using the $args collection. $ComputerName (a named parameter) is a lot more meaningful than $args[1].

• We hate inconsistency. In whatever you write, try to be consistent with what’s already in PowerShell. A parameter named –ComputerName is great; a parameter named –ComputerNames is not great. Nothing else in PowerShell uses the plural, so you shouldn’t either.

• We love scripts and commands that output objects and not formatted text. That means you shouldn’t embed a Format cmdlet in your script, nor should you use Write-Host in most cases. Exceptions are made by the name of your script or command: we’d expect a command named Show-Info to show information on the screen, implying Write-Host is being used. A command named Get-Something, on the other hand, should output unformatted objects.
We’ll pipe those to a Format command ourselves if we want them formatted.

- We hate scripts that are hard to read. With scripts (as opposed to one-liners), focus on neat formatting. Avoid the backtick (`) as a line continuation trick because that little bugger is hard to see.

- We love elegance. For example, a lot of folks think the expression "$computer cannot be reached" is easier and less complex than ("{0} cannot be reached" –f $computer) even though those two expressions produce the same result. Both of those are prettier than ($computer + " cannot be reached"). Just keep that in mind.

- We hate redundancy... well, not in server farms, but definitely in code. For example, in the command `Get-Service | Select-Object –Property @{n='Name';e={$PSItem.Name}}` – is all that really needed? Couldn’t you just go for `Get-Service | Select-Object –Property Name` and get the same effect with less typing? We’re not saying this kind of thing will land you on a judges Worst list, but for some judges we know it’ll keep you off their Best list.

- We love functions that use `[CmdletBinding()]` and take advantage of the features it enables. For example, if you’re manually setting something like `$VerbosePreference` or `$DebugPreference` at the top of your script... then you’re missing the point.

- We hate scripts that pollute the global scope. It’s not yours. We don’t track mud into your house, don’t litter up our global scope with your variables. Exception: if you write a script module that politely adds global “preference” variables, and then politely removes them when we unload it, that’s cool.
What Not to Over-Obsess About

In some event scenarios, we'll give you an example of the expected output. It's an example, not a mandatory deliverable. Basically, so long as you have all the right property names, and the values look legitimate, you're fine. They don't need to be in the order we list them, and they don't need to exhibit the exact same formatting, unless the event scenario calls out a specific requirement. In many cases, the example output is formatted to fit in a PDF – it's gonna look different in the PowerShell console, and we're cool with that.

However, make sure you meet the scenario requirements. If a requirement says to display a value in gigabytes, you'd better do it. Expect your CrowdScore to be pretty low if you display in bytes when the scenario explicitly asked for gigabytes.

Let's be clear on something: if your goal going into the Games is to get on every judge's Best list... then you're playing for the wrong reasons. If your goal is to get an amazing CrowdScore in every event... you're probably going to be disappointed. This is a learning event.

Here's another way to think about it: you're going to have your script peer reviewed by dozens of people, and possibly get some expert feedback from some of the biggest names in the industry. That alone is worth participating. The prizes are just icing on the cake!

DO NOT treat the previous list of guidelines as some kind of Secret Checklist for Winning. It isn't. They're pretty good ideas in general, but they're not the only good ideas, and like all rules they come with their own exceptions.

DO obsess about finding a clever, elegant, well-written solution to each problem. Knock our socks off!
Try Not to Miss the Whole Point of the Games

You’ll notice that the only score you’ll receive is from your peers in the community – and we can’t tell them how to score you. Heck, some of them may not feel they’re even qualified to hand out scores, and a few of those might be right (more on that in a moment).

The score is nice... but it isn't the point of the Games. The point is to think of creative approaches to real-world problems... and to implement those approaches as best you can. That’s why our expert judges will be picking their Best And Worst lists... to highlight creativity, attention to detail, and overall sk1llz. Every judge will have different opinions, preferences, and techniques... and they’re all purely subjective. We’re not giving them any guidelines whatsoever. So there’s no Secret Checklist to Winning.

We are making our judges blog about what they like and don’t like... and that is the real point of the Games: to learn.
Why You Should Vote on Entries, Even if You Think You’re Not Qualified

Well, actually, you are qualified. Just ask yourself: “is this a script or command I’d want running on my production environment? Is this the work of a person I’d hire, if I had the opportunity? Did I learn something from this entry?” And then vote with your heart. Everyone is qualified. And if you can leave a brief comment about why you voted the way you did, even better – votes are anonymous, as are entries (during the voting period), so just be polite and professional and treat others as you’d want to be treated yourself.

And remember, every vote is a prize raffle ticket!

Oh, and this should go without saying, but we’re gonna say it anyway: don’t be mean. We do have systems in place to watch for odd voting patterns, like handing out all 1s or 5s just to rack up Pointlets. We also watch for sequence patterns and other signs of abuse. All of those things trip alarms, and we look into things manually too. If we find wrongdoing, you’ll be banned from the Games for life. Seriously. Oh, we’ll talk to you about it first – we’re not mean – but we absolutely won’t stand for this system being abused.

The future of the Games is in peer review and voting. Your opinion – the opinion of someone working in a production environment – is what’s important in the real world, not the opinion of some fancy-pants judge. Part of the Games – the expert judge commentary, for example – will make you a better voter/judge in the future, and that’s how we’re going to help build a better overall PowerShell community. So respect the vote.
Thanks to Our Presenting Sponsors

Their generous support makes the Games and all of PowerShell.org possible. Please offer them your support and thanks.

GOOD LUCK!