## #Slide - <Webinar Title>

So this is "Simplify automation of common file transfer tasks for secure business partner integration" presented by IPSwitch. In today's webinar, we'll be covering a typical problem that may exist in your environment today; we've got too much data, in too many places, in too many formats that needs shared. We'll go over the problem of all this disparate data and then at the end we'll go over a really cool demo where I'll be using PowerShell to pull data out of a Microsoft Azure SQL database and converting to a XML file all while using MOVEit to manage it.

I hope by the end of the webinar, you'll have a hint at how you can overcome this problem of disparate data using PowerShell and MOVEit Central.

## #Slide - About the Presenter

## #Slide - Data is Everywhere

In today's business struggle to remain competitive, we sometimes experience what I like to call the data pandemic. Data is good, right? We all love data. The more data the better. How does that old saying go? Knowledge is power and in today's currency, knowledge is measured in data. However, what happens when you've got sources of data popping up all over the place like a zombie apocalypse?

Everything generates data; applications, the services that those applications depend on and the operating system that those services depend on. Unfortunately, this leads to data being spread out everywhere. To think that we've got data stored in hundreds of places in our own datacenters, someone had the bright idea to start storing data in someone ELSE'S datacenters. Can you say "cloud"? Although cloud services are a huge leap forward in reduced cost and worker productivity it doesn't make the job of a data wrangler any easier.

## #Slide - Data is Represented in Lots of Ways

To make matters worse, not only is data spread out everywhere it's not the same *kind* of data. Your data is represented in numerous formats. You've probably got critical data stored in huge data warehouses using Microsoft SQL Server or Oracle. You might have data stored in MySQL databases or even non-relational databases like Hadoop or MongoDB. On top of just databases, you probably also have critical data stored in flat files represented by XML or JSON, for example. You probably also have Active Directory too. This is all data and it's important in some way to your business.

Although managed by themselves it's not a problem but when you begin to try to mix and match these formats is when things can get ugly. Do you know how many products are out there that are built just to sync Active Directory data with some other system? On my first Google search, I counted 8 different systems and that was just one the first two pages of results!

## #Slide - What if You Need to Share that Data?

Companies know that many businesses have disparate data spread out everywhere. It's a common problem to have data stored in Application X when Application Y or business partner Z needs to consume that data in a different format. This is why you see so many software products built just for the sole purposes of syncing different data formats. But what if I told you you don't really need any of these products? You just need a little scripting ingenuity and a product that can deliver the data from point A to point B securely and without tripping up your auditors.

Sharing this disparate data across businesses becomes difficult. Your business partner needs to import the data in a completely different format than yours. There's no way to simply export out a SQL database, for example, to something like XML. That's gonna require some work. Without some kind of intermediary it's simply not going to happen.

Both businesses need a standard format in order to share information back and forth.

## #Slide - Who's Going to Change the Format?

Let's play pass the buck. I'm sure you've all been there. Whether it's you and your coworkers in your own dept, your dept and a vendor you use or even when it's between two IT departments working for different yet collaborating companies. Two parties have a mutual problem; one party has data format A and one party has data format B. Although perfectly fine by themselves, a problem arises when they need to share this data with each other. Who's going to cave? Who's going to be the one to convert *their* data format to the *other* party's data format? Is it going to be the MySQL company that's forced to convert their data to XML for the company that stores a bunch of critical data in a bunch of XML files or the other way around?

If you're watching this webinar, I probably know the answer to that question already. I'm sorry. You probably have a collection at home of short sticks, don't you? Oh well. It's just another great learning opportunity to learn new products and advance your career! That's what I always say…..usually.

In any case, it's up to you now to figure out a way to securely deliver your data to your business partner in an efficient, automated fashion. Lucky for you, I know of a company or two that can help you with that.

## #Slide - The Task at Hand

You've got a journey ahead of you that requires 3 rough steps. The first is data retrieval. This is a matter of simply getting at the data; just pulling it out in it's raw form. Although you may think; that's no problem. For Active Directory, I just use Active Directory Users and Computers, for SQL Server I use SQL Server Management Studio, for XML, I'll just fire up my text editor, etc. However, you need to scale this up. You need to think automation. Your business partner needs this data bad and needed it yesterday…every day or whenever new data is available. Your GUI tools aren't going to cut it. It's time to bring in the big guns; Windows PowerShell. By using PowerShell, you are able to easily query just about any data source you have and have it stored in a way that allows you to easily move it along in it's journey to becoming a new format.

Next, is the conversion phase. If you've used PowerShell to gather the information, it's now ripe for conversion. If you know the source data schema and the destination data schema plus the rules that dictate 1:1 match between both sources you can develop scripts to quickly convert source A to destination B.

Finally, you'll need something to securely and efficiently deliver the data. MOVEit is a large product that can do a lot of things but one of the features I like the most is it's ability to watch a folder and act on it. In our case, PowerShell will query and convert the data and MOVEit will be tasked to pick it up, wrap it up into a nicely encrypted package and securely deliver to it's intended destination. This is why I like to call MOVEit the UPS of file delivery.

## #Slide - Getting at Your Data

The first step in your task of delivering disparate data to your business partner is first being able to pull the source data out. This is probably an ongoing process so you don't want to have to depend on someone going in and manually pulling the data. The process has to be automated.

Next, is has to be flexible. Things change such as where the data is located, the methods required to get to it and various other things like database name changes, table changes, etc. You need a way that you can easily change to adapt to these changes. Nowadays, you can manage just about any data source with PowerShell. This is why it's such a great choice here.

## #Slide - Converting Your Data to Meet Partner Criteria

Once you've managed to get your data out of whatever data source it's in it's now a matter of converting that into the format of the destination. The actual nuts and bolts to do this are highly dependent on the source and destination data. However, there are a few general rules you must follow when starting down this path.

The first is to know both schemas. Every data source has a schema. A schema is particular structure the data must be represented in. For example, you might be most familiar with a database table schema. It is a logical grouping of columns and rows that will always adhere to the schema they are a part of. For example, you only have a certain number of rows and columns in a database table, an Excel spreadsheet or a CSV file. A schema dictates this. Before you start converting the data you must know how each data source is structured.

Next, you must adhere to what I call conversion rules. These are rules that allow you to match up particular rows of data in a 1:1 format or simply that we're comparing apples to apples. Maybe we've got a MySQL database table with the columns; FirstName and LastName. We'd like to convert this database to XML. We then need to know how to represent the first name and last name in that XML file. Since XML is a completely different schema, we'll have to come up a way to represent that data differently. In the case of XML, we might choose a particular namespace and we'd also need to decide on a parent tag to represent all of the rows from the MySQL table.

In my case, I develop these rules with Windows PowerShell.

## #Slide - Sending Data Securely Across Trusted Networks

At this point, our data is in our business partner's proper data format and it's ready to be sent. We now move the process from PowerShell to MOVEit Central. To put it to a bowling saying, "PowerShell set the pins and MOVEit will knock 'em down". This is where simply using PowerShell for this task doesn't make sense. Even as a PowerShell MVP, I'm not about to try to code up another MOVEit Central. It'd take years! No thank you. I'm just going to hand the baton over to MOVEit now.

Once the data is sitting in a folder, MOVEit can now pick it up and securely deliver it to our business partner. We can now trust that our business partner will receive the data intact and will be securely delivered in a number of different ways. MOVEit can automatically check the folder, send the data (optionally encrypt it) and all the while providing accessing to reporting (One thing that doing this in PowerShell would take a long time to implement).

## #Slide - PowerShell: MOVEit's Sidekick

PowerShell is only a tool. It does a lot of things well but has its limits. PowerShell requires a skilled hand to build all of this code, requires time to build it and sometimes it's not manageable for a non-developer. PowerShell did a great job at reading and converting the data we needed to send but it needed MOVEit to securely get that data to our business partner. MOVEit is great at securely and efficiently transferring files to meet compliance regulations. PowerShell…well…it'll usually get the job done but I'm not guaranteeing it wouldn't take you years to build a system like MOVEit Central let alone please the auditors.

Not all data is treated equally. There are times when we need to move data from point A to point B with MOVEit but need another step. This is where PowerShell shines. PowerShell can be thought of as a shim between the raw data and MOVEit Central. It's not the solution. The solution is the teaming up of PowerShell and MOVEit Central. MOVEit is the star here with it's fancy auditor-pleasing security features, reporting and so on. PowerShell's just the sidekick.

Robin doesn't drive the batmobile; that's Batman's job. But, without Robin, Batman would probably get lost. We guys don't stop and ask for directions, you know? As you've seen, by using PowerShell as a complement to MOVEit really takes file transfer to the next level.

# #Slide - Building PowerShell Scripts for MOVEit Central

Using your PowerShell scripts with MOVEit is pretty simple. You just have to remember to include a few optional abilities in your script and know how to execute them with MOVEit. If this doesn't make sense now, we'll go over a quick demo in a minute where you'll see exactly what I'm talking about.

First, you'll need to ensure your script has a few optional abilities. First of all, if you need your script to process a file that MOVEit is using, you'll need to include a parameter that allows MOVEit to pass in a file path for your script to pick up and handle. Next, you'll need to use the Write-Output cmdlet to send status or logging messages to the standard out stream in order for your messages to show up in the MOVEit debug log. ..and as a last step, MOVEit uses exit codes to determine success or failure. Using PowerShell, you can set custom exit codes depending on different criteria. I'll have examples of each of these in the demo.

You'll also need to know *how* to execute your scripts in MOVEit. First, you will execute your PowerShell scripts with the Command Line App process. This is a general task process which you can specify the powershell.exe file and then pass it your script name and any parameters you might have. Finally, you'll also need to be aware of MOVEit's StdOut, StdError and InputFile variables. You'll use these variables inside of your Command Line App parameters to pass files back and forth and to get your script output to the debug log.

# #Slide - Demo: Azure SQL DB/PowerShell/MOVEit

Now that you've got a sense of what we're dealing with let's put our tools into action and show you an example. In the demo, I'm going to show you an example scenario of what we've been talking about. In this example, we'll have PowerShell pull data out of a Microsoft Azure SQL database, store that dataset and convert it to XML in memory and then finally create a XML file from that.

I'll go over the PowerShell code to make it happen and then Ipswitch will demonstrate how to setup the script in MOVEit Central and how to properly execute it.