Powershell and the M&M SOAP API

Overview

Powershell can be used to interact with the Men and Mice SOAP API. The mmSoap.ps1 script that is attached to this article will simplify the process greatly and make a robust SOAP client.

The current version of mmSoap.ps1 is **version 9.2.2 updated 06/03/2019**

You can retrieve the script version the Powershell way:

```
>Get-Help .\mmSoap.ps1
or
>man .\mmSoap.ps1
```

The version information should be listed up in the Synopsis section of the help text. If not please consider to test and update your mmSoap.ps1 script as it's most likely outdated.

To initialize a SOAP client on your machine, download mmSoap.ps1, and simply enter:

```
>.\mmSoap.ps1
```

To connect to https instead of http just add the -https switch:

```
>.\mmSoap.ps1 -https
```

If you have version 6.6 or later of the Men & Mice, you can use the JSON implementation for the transport, which is more lightweight. To do that simply add -json:

```
>.\mmSoap.ps1 -json
```

mmSoap.ps1 will create a library of helper functions, **mm<MethodName>**, and make them available in the open Powershell session. mmSoap.ps1 also creates helper functions, named **New-mm<ObjectName>,** to assist with creating the various objects in the M&M API. This library of helper functions, along with the tab autocompletion of function parameter names in Powershell, makes it very easy to work with the M&M SOAP API.

When mmSoap.ps1 is invoked, it remembers the last used connection information, including the username and password (stored encrypted). Therefore, it is usually sufficient to call mmSoap.ps1 without parameters after calling it the first time and providing the connection information.

To list the available parameters for mmSoap.ps1, you can either use the parameter autocomplete by adding a hyphen and using the tab-key, or via man .\mmSoap.ps1:

```
>man .\mmSoap.ps1
mmSoap.ps1 [-json] [[-mmWeb] <String>] [[-mmCentral] <String>] [[-username] <String>]
[[-password] <String>] [-singleSignOn] [[-namespaces] <Object>] [[-addressSpace]
<String>] [-https] [-http] [[-port] <Int32>] [-useWebServices] [-useAuthHeaders]
[[-timeoutSeconds] <Int32>]  [-quiet] [-askForWebCredentials] [-useDefaultCredentials]
[-regenerate] [-reloadClient] [-interactive] [-clearDefaults] [-printResponse]
[-measureCommands] [[-defaultsFile] <String>] [<CommonParameters>]
```

To get the list of DNS zones, the user simply types:

```
> mmGetDNSZones -filter "name:^somezonename.com"  -limit 4
```

If you forget to assign the result to a variable, the last results are stored in $mmLastResult, which is a global variable in the Powershell session. The zone retrieved above will therefore now only be accessible in the $mmLastResult variable.

Creating complex types and arrays is achieved by using helper methods with the naming convention New-mm<Object>. For instance, to create a Property object and assign values, use:

```
> $prop = New-mmProperty -name 'Monitored' -value $true
```

and to create an ArrayOfProperty with the property value(s):

```
> $propArray = @($prop)
```

Examples

Example 1.

Add a DNS record to a zone

```
> mmAddDNSRecord -dnsRecord (New-mmDNSRecord -name testrecord -type A -ttl 7200 -data
10.1.2.3 -dnsZoneRef 'example.com.' -enabled $true) -saveComment 'This is a test'
```

When the DNS Zone has an ambiguous name, i.e. multi-master zones, then the dnsZoneRef can be in the format '<DNSServer>:<View>:<ZoneName>', where <View> name refers to views on BIND servers, but should be empty in all other cases.

```
> mmAddDNSRecord -dnsRecord (New-mmDNSRecord -name testrecord -type A -ttl 7200 -data
10.1.2.3 -dnsZoneRef 'dns1.example.com.::example.com.' -enabled $true) -saveComment
'This is a test'
```

If the DNSZone that the record should be added to has been retrieved, it's unique reference should be used instead:

```
> $recRef = (mmAddDNSRecord -dnsRecord (New-mmDNSRecord -name testrecord -type A -ttl
7200 -data 10.1.2.3 -dnsZoneRef '{#4-#32124}' -enabled $true) -saveComment 'This is a
test').ref
```

Example 2.

Add a DNS Zone

```
> $zoneRef = (mmAddDNSZone -dnsZone (New-mmDNSZone -name 'soaptest.com.' -dnsViewRef
'dns1.example.com.:' -type Master) ).ref
```

Here dnsViewRef has been provided in the format <DNSServer>:<View>, and since myserver.example.com. does not have views or is not a

BIND server, <View> is empty.

To add a slave zone, the masters parameter is required

```
> mmAddDNSZone -dnsZone (New-mmDNSZone -name 'soaptest.com.' -dnsViewRef
'dns1.example.com.:' -type Slave) -masters '12.34.56.77'
```

Example 3.

Create a Subnet/Range

```
> mmAddRange -range (New-mmRange -name '192.168.1.0/24' -subnet $true
-customProperties @((New-mmProperty Title 'Test 192 subnet'),(New-mmProperty
Description 'Test description'))
```

Example 4.

Retrieve the next free IP address within a particular IP address Range

```
> $address = (mmGetNextFreeAddress -rangeRef 192.168.1.0/24 -startAddress
192.168.1.100 -ping $true).address
```

This will retrieve the next free IP Address within 192.168.1.0/24 that is above 192.168.1.100. The IP address must not respond to ICMP ping.

To retrieve the next free address and claim it temporarily at the same time (to prevent concurrency/race conditions), the temporaryClaimTime (seconds) parameter can be provided

```
> $address = (mmGetNextFreeAddress -rangeRef 192.168.1.0/24 -startAddress
192.168.1.100 -ping $true -temporaryClaimTime 100).address
```

Note that the IP will be released automatically 100 seconds after this command is invoked, so the IP must be permanently claimed, shown in example below.

Example 5.

Claim an IP address

```
> mmSetProperties -ref '1.2.3.4' -objType 'IPAddress' -properties @((New-mmProperty
'claimed' $true)) -saveComment 'This is a test'
```

Example 6.

Duplicate a DNS Zone

```
> $templateZone = (mmGetDNSZone -dnsZoneRef template.zone.com.).dnsZone
> $templateRecs = (mmGetDNSRecords -dnsZoneRef $templateZone.ref).dnsRecords
> $templateZone.name = 'newzone.example.com.'
> $theNewZoneRef= (mmAddDNSZone -dnsZone $templateZone).ref
> foreach ($rec in $templateRecs) {
> $rec.dnsZoneRef = $theNewZoneRef
> }
> $result=mmAddDNSRecords -dnsRecords $templateRecs
```

See this article for information on how to run Powershell scripts through Men & Mice Central