# Lync AutoOps v.1.5.0 Admin Guide

*Author: Max Sanna*

*Email: max(at)maxsanna.com*

*Date: 12/01/2015*

*Version: 1.5.0*

Contents

[Lync AutoOps v.1.5.0 Admin Guide 1](#_Toc408842175)

[Introduction – Why Automate Lync User’s Lifecycle? 2](#_Toc408842176)

[What does the script do? 2](#_Toc408842177)

[Infrastructure Prerequisites 3](#_Toc408842178)

[Installation Instructions 3](#_Toc408842179)

[Configuration Parameters 4](#_Toc408842180)

[Recipients List 6](#_Toc408842181)

[Sample Supported Topologies 6](#_Toc408842182)

[Running LyncAutoOps unattended 9](#_Toc408842183)

## Introduction – Why Automate Lync User’s Lifecycle?

This is a topic I found myself having to face during many deployments: how will the customer’s IT department eventually manage joiners/movers/leavers in Lync once we finish the deployment?

When dealing with very big deployments, this is usually not an issue, as some kind of IDAM (IDentity and Access Management) system will be already present (such as Forefront Identity Management, or Dell Identity Manager, etc.). If that’s the case, all I do is normally provide the IDAM team with the one-liners to manage Lync users and let them know at which point of their provisioning workflows they’ll fit best.

The problem instead arises for pretty much any company below 5,000 users in my experience, as they’ll normally provision AD groups and Exchange mailboxes using independent PowerShell scripts (or \*gasp\* by hand every time!).

LyncAutoOps aims at solving this issue for all these companies without a proper IDAM system.

The script originated as a bunch of individual scripts tied up together with a batch file, not pretty at all, which I had to develop from scratch for a particular customer. Upon noticing I had to use and re-use them for so many deployments, I decided to tidy things up and make a unified script, reusing functions etc. I also moved all the configuration parameters in an easy to read (kind of, if you’re into XML) config file, which makes the job of implementing the script extremely quick.

## What does the script do?

Essentially the script will perform 4 functions, which are the basics of Lync user management:

* Enable Lync users who are members of an Active Directory group
  + As of version 1.5.0, the script is now pretty flexible and can support different topologies. It can either work in ‘simple’ mode and enable users either on a single pool or on a paired pool, balancing the users 50/50 on the two different pools, or it can work in the new MultiPool mode, which supports multiple pairs of Lync pools, with multiple locations assigned to them, and uses an AD attribute to identify where the user resides and on which pool to enable them.
  + After enabling the users, it will assign a number of Lync policies to all the users.
  + It will also remove the users from the AD group used for activations, so the group stays empty after usage.
* Suspend Lync users who are **enabled** in Lync but **disabled** in Active Directory. I saw many customers not realising that simply disabling someone in Active Directory doesn’t prevent them from using Lync at all, because Lync issues a client certificate that lasts 180 days. Therefore, after you disable a user in Active Directory, chances are he will be able to still use Lync on a previously-used device for another 6 months. Hardly security-conscious, is it?
* Reactivate Lync users who are **disabled** in Lync but **enabled** in Active Directory. This does the opposite: an employee may have gone in maternity for a few months, the IT department disables her account, then she comes back and gets it re-enabled. The script will re-enable these people automatically.
* Delete Lync users who are **disabled** in Active Directory and who **haven’t logged in** for a custom number of days (usually 90 is a good number). This allows the Lync team to keep the backend database tidy by removing all the users who have left the company. It’s always good practice to delete people from Lync before deleting the AD object, or a lot of ‘leftovers’ will remain in the Lync backend database.

On top of all this, it also emails a custom list of recipients the results of each script run. If there were errors, a separate log file containing only the errors will also be produced to simplify remediation.

## Infrastructure Prerequisites

Even if I could have saved myself a fair bit of work by using the Active Directory Powershell module, I decided to use Powershell 1.0-style .NET objects to perform the AD-related operations. The reason for this is that in quite a lot of environments I’ve worked in, they either had Windows Server 2003 or 2008 R1 for domain controllers, or they had 2008 R2 but they disabled the AD Powershell. For one reason or another, as much as I’d wish all customers to migrate their DCs OS and forest functional level to the latest and greatest on the day it comes out, this situation keeps on coming up, so I decided to avoid the problem entirely and not rely on AD Powershell.

Another possible workaround I was thinking of was to use the good old Quest ActiveRoles Management Shell module, but I didn’t for two reasons: first, some companies don’t really want to install 3rd party modules onto their servers without a really good reason, and second, Dell just decided around the end of November 2014 to put the module (which was free under Quest since forever) behind a paywall for ActiveRoles server customers. Thanks Dell :-/

As outlined above, the only requirements for the script to run are:

* Microsoft Lync Server 2013 Core Components installed on the server on which LyncAutoOps will run
* PowerShell 3.0 or above (it should work with PowerShell 2.0 as well but I haven’t tested it)
* PowerShell signing set to Unrestricted. I’m planning to sign the script at some point though.
* A user with enough rights to operate on AD and on Lync users (normally CSUserAdministrator would be enough, but for Powershell operations I found out more rights are required, so I tend to give CSAdministrator rights to the service account which will run the script)
* A new AD group in the domain which contains the users to enable, it can be called like in the script: ‘**NewLyncUsers**‘, but it doesn’t really matter. This is the group that the Operations team will have to remember to add new users to.
* If you’re going to use the MultiPool mode you need to decide an AD attribute which will be used to look up the user’s location. By default the script will use ‘**physicalDeliveryOfficeName’** (i.e. the Office attribute in Active Directory Users & Computers).
* Windows 2003/2008/2008 R2/2012/2012 R2 Active Directory
* **All the users to enable should already possess an email address.** This part is important – if they don’t have an email address already provisioned, the activation will fail, because we’re using the email as a base to compose their sip address. Should this be your situation, you will have to modify the script where it enables users, and use another criteria to generate their sip address. Please read the documentation for Enable-CsUser, especially around the -SipAddressType parameter, in order to find a viable alternative for your situation. Perhaps using the UPN? :)

## Installation Instructions

In order to install the script easily, I recommend to unpack the contents of the ZIP file (you’ll find it at the end of the article) in the C:\ root on a server with Lync 2013 Core Components already installed. Normally, one of the Lync Server 2013 frontends will be a perfect candidate.

### Configuration Parameters

In the folder, there is a file called LyncAutoOps.config, which contains all the configuration parameters required by the script. The file is commented, so it shouldn’t be difficult to fill in all the parameters for the environment in use, although we’ll provide further examples later in the documentation.

**IMPORTANT:** **If you were already using an older version of LyncAutoOps, you’ll need to use this new configuration file as the schema has changed, therefore copy the old attributes from the previous version into the new one.**

<?xml version="1.0" encoding="utf-8"?>

<Config>

<ADSettings>

<!--sAMAccountName of the group containing new users to enable-->

<NewLyncGroup>NewLyncUsers</NewLyncGroup>

<!--Netbios domain name-->

<NetBiosDomain>CORP</NetBiosDomain>

<!--Search root domain expressed in DN format-->

<DNDomain>DC=corp,DC=contoso,DC=com</DNDomain>

<!--AD Attribute containing the user location - only required when PoolTopology is set to MultiPool-->

<UserLocation>physicalDeliveryOfficeName</UserLocation>

</ADSettings>

<LyncSettings>

<!--Setting PoolTopology to Simple will only enable users on the first Lync Pool pair specified in the array. If two pools

are specified, the script will enable users 50/50 on both pools.

Setting PoolTopology to MultiPool will enable users in different single pools or pool pairs as specified by the AD Attribute

entered in UserLocation. When using this modality, each pool in the array needs to contain the location name for the script

to identify where to enable users.-->

<PoolTopology>MultiPool</PoolTopology>

<!--Lync Pool Array-->

<!--For each pool location, duplicate a <Pool> block, set its location, and one or two FQDNs for the pair if available.

The pool set with IsFallBackPool to true, will be used in case the user to enable doesn't have a location set.

You can add multiple Location attributes to a single pool if you want to use it for multiple user locations.

Delete the Pool blocks you don't need in your deployment.-->

<PoolArray>

<Pool>

<Location>London</Location>

<Location>Manchester</Location>

<FirstPoolFQDN>londonpool01.contoso.com</FirstPoolFQDN>

<SecondPoolFQDN>londonpool02.contoso.com</SecondPoolFQDN>

<IsFallBackPool>True</IsFallBackPool>

</Pool>

<Pool>

<Location>New York</Location>

<FirstPoolFQDN>nycpool01.contoso.com</FirstPoolFQDN>

<SecondPoolFQDN>nycpool02.contoso.com</SecondPoolFQDN>

<IsFallBackPool>False</IsFallBackPool>

</Pool>

<Pool>

<Location>Chicago</Location>

<FirstPoolFQDN>chicagopool01.contoso.com</FirstPoolFQDN>

<SecondPoolFQDN></SecondPoolFQDN>

<IsFallBackPool>False</IsFallBackPool>

</Pool>

</PoolArray>

</LyncSettings>

<!--Settings these values to False will disable portions of the script's actions-->

<ScriptFunctions>

<!--This portion enables users members of NewLyncGroup for Lync-->

<Enablement>True</Enablement>

<!--This portion suspends Lync users who are disabled in AD but enabled in Lync-->

<Suspension>True</Suspension>

<!--This portion reactivates Lync users who are enabled in AD but suspended in Lync-->

<Reactivation>True</Reactivation>

<!--This portion deletes Lync users who are disabled in AD and whose lastLogonTime is more than the number of days specified in the parameter DeleteThreshold-->

<Deletion>True</Deletion>

<DeleteThreshold>90</DeleteThreshold>

</ScriptFunctions>

<Notifications>

<!--Settings this parameter to False will disable daily email notifications of the script's operations-->

<EnableNotifications>False</EnableNotifications>

<!--This is the email address from which notifications will be sent-->

<FromAddress>lyncserver@contoso.com</FromAddress>

<!--SMTP server to use - make sure you have rights to send email impersonating another user if using Exchange-->

<SMTPServer>mail.contoso.com</SMTPServer>

<!--Path to the recipient list - add email addresses one after another on separate lines-->

<RecipientsPath>.\Recipients.txt</RecipientsPath>

</Notifications>

<!--To assign a global policy, set any of these variables to empty, otherwise add the name of the policy to assign without the Tag: prefix-->

<UserPolicies>

<ClientPolicy>TestPolicy</ClientPolicy>

<ConferencingPolicy>BigConferences</ConferencingPolicy>

<ExternalPolicy></ExternalPolicy>

<PinPolicy></PinPolicy>

<ArchivingPolicy></ArchivingPolicy>

<!--To enable archiving in Microsoft Exchange rather than in Lync's own SQL Instance, set this parameter to True (requires integration with Exchange 2013)-->

<ExchangeArchiving>False</ExchangeArchiving>

</UserPolicies>

</Config>

Here’s a list of all the parameters, with their explanation:

* **ADSettings**
  + **NewLyncGroup**: sAMAccountName of the group containing new users to enable
  + **NetBiosDomain**: Netbios domain name containing the users to enable
  + **DNDomain**: Search root domain expressed in distinguished name format
  + **UserLocation**: AD attribute name containing the user’s location
* **LyncSettings**
  + **PoolTopology**: This can be set to **Simple** to enable users on the first Pool found below (or pair of pools) regardless of users’ locations, or to **MultiPool** to use multiple pairs of pools and enable users based on their location
  + **PoolArray**
    - **Pool**: these are the block containers for each pool pair
      * **Location:** Enter the name of the location as it appears in the users’ location field. You can add as many location attributes as desired, so a pool can be used for multiple locations
      * **FirstPoolFQDN:** Enter the FQDN of the first pool for this location
      * **SecondPoolFQDN:** (Optional) Enter the FQDN of the paired pool for this location. If left blank, users will be activated only on the first pool, so it’s suitable for when there is no pair or the paired pool is a hot-standby one without users homed on it
      * **IsFallBackPool:** If set to True, this pool will be used to activate users with misspelled or missing location attributes. If set to false, these users will fail and an error will be logged. Only one pool can have this attribute set to True.
* **ScriptFunctions**
  + **Enablement**: if set to true enables users for Lync
  + **Suspension**: if set to true suspends users for Lync when disabled in AD
  + **Reactivation**: if set to true reactivates users for Lync when enabled in AD
  + **Deletion**: if set to true deletes users from Lync when disabled in AD and beyond the last logon threshold
  + **DeleteThreshold**: number of days since last logon for deletion
* **Notifications**
  + **EnableNotifications**: if set to true it emails the recipients contained in Recipients.txt with the results of the script run
  + **FromAddress**: email address to use when sending out notifications
  + **SMTPServer**: SMTP server FQDN, normally a local Microsoft Exchange
  + **RecipientsPath**: path to the list of recipients to be used
* **UserPolicies**
  + **ClientPolicy**: name of the client policy to assign
  + **ConferencingPolicy**: name of the conferencing policy to assign
  + **ExternalPolicy**: name of the external access policy to assign
  + **PinPolicy**: name of the PIN policy to assign
  + **ArchivingPolicy**: name of the archiving policy to assign
  + **ExchangeArchiving**: when set to true it sets new users to archive IMs in Exchange 2013 mailboxes

### Recipients List

If you choose to receive the script notifications via email, you should modify the file Recipients.txt, and add all the email addresses of the people who will receive the notifications, one after another on separate lines.

## Sample Supported Topologies

The next chapter will show some supported topology examples, to give a better understanding of how to configure the script for different scenarios.

#### Single Lync Pool



For this simple scenario, we’ll set the script’s PoolTopology to Simple and just enter the first FQDN in FirstPoolFQDN, Location and IsFallBackPool can be left in place but won’t really be used during script execution.

<LyncSettings>

<PoolTopology>Simple</PoolTopology>

<PoolArray>

<Pool>

<Location>London</Location>

<FirstPoolFQDN>lyncpool01.contoso.com</FirstPoolFQDN>

<SecondPoolFQDN></SecondPoolFQDN>

<IsFallBackPool>True</IsFallBackPool>

</Pool>

</PoolArray>

</LyncSettings>

#### Paired Lync Pool



Again, a very simple scenario. The only difference from above is that we add also a SecondPoolFQDN so that users will be enabled 50/50 on both pools.

If for some reason you only want to keep users on the first pool, while the second stays empty in hot-standby, then use the configuration from the ‘Single Lync Pool’ paragraph and don’t enter a second pool FQDN.

<LyncSettings>

<PoolTopology>Simple</PoolTopology>

<PoolArray>

<Pool>

<Location>London</Location>

<FirstPoolFQDN>lyncpool01.contoso.com</FirstPoolFQDN>

<SecondPoolFQDN>lyncpool02.contoso.com</SecondPoolFQDN>

<IsFallBackPool>True</IsFallBackPool>

</Pool>

</PoolArray>

</LyncSettings>

#### Sample SBA Topology



In this particular scenario, we have a single pool in London, with 4 SBAs attached to it. In such a scenario, we’ll just set up five pool entries, each one with their location field set up. We’ll set the main London pool as IsFallBackPool so that users without an explicit location will be homed there by default. If one of the sites is supposed to home other locations as well, simply add another Location line to the pool block. If the London pool was paired, simply add the SecondPoolFQDN and the script will enable people 50/50 on it.

<LyncSettings>

<PoolTopology>MultiPool</PoolTopology>

<PoolArray>

<Pool>

<Location>London</Location>

<FirstPoolFQDN>londonpool01.contoso.com</FirstPoolFQDN>

<SecondPoolFQDN></SecondPoolFQDN>

<IsFallBackPool>True</IsFallBackPool>

</Pool>

<Pool>

<Location>Guildford</Location>

<FirstPoolFQDN>guildfordsba.contoso.com</FirstPoolFQDN>

<SecondPoolFQDN></SecondPoolFQDN>

<IsFallBackPool>False</IsFallBackPool>

</Pool>

<Pool>

<Location>Chelmsford</Location>

<FirstPoolFQDN>chelmsfordsba.contoso.com</FirstPoolFQDN>

<SecondPoolFQDN></SecondPoolFQDN>

<IsFallBackPool>False</IsFallBackPool>

</Pool>

<Pool>

<Location>Brighton</Location>

<FirstPoolFQDN>brightonsba.contoso.com</FirstPoolFQDN>

<SecondPoolFQDN></SecondPoolFQDN>

<IsFallBackPool>False</IsFallBackPool>

</Pool>

<Pool>

<Location>Edinburgh</Location>

<FirstPoolFQDN>edinburghsba.contoso.com</FirstPoolFQDN>

<SecondPoolFQDN></SecondPoolFQDN>

<IsFallBackPool>True</IsFallBackPool>

</Pool>

</PoolArray>

</LyncSettings>

#### Multiple Global Pool Pairs



In this example, we have multiple locations, some of which with pool pairing, some of which serving multiple cities. Again, the configuration is pretty straightforward:

<LyncSettings>

<PoolTopology>MultiPool</PoolTopology>

<PoolArray>

<Pool>

<Location>London</Location>

<Location>Manchester</Location>

<Location>Cardiff</Location>

<FirstPoolFQDN>londonpool01.contoso.com</FirstPoolFQDN>

<SecondPoolFQDN>londonpool02.contoso.com</SecondPoolFQDN>

<IsFallBackPool>True</IsFallBackPool>

</Pool>

<Pool>

<Location>New York</Location>

<Location>Hoboken</Location>

<FirstPoolFQDN>nycpool01.contoso.com</FirstPoolFQDN>

<SecondPoolFQDN>nycpool02.contoso.com</SecondPoolFQDN>

<IsFallBackPool>False</IsFallBackPool>

</Pool>

<Pool>

<Location>Chicago</Location>

<FirstPoolFQDN>chicagopool01.contoso.com</FirstPoolFQDN>

<SecondPoolFQDN></SecondPoolFQDN>

<IsFallBackPool>False</IsFallBackPool>

</Pool>

</PoolArray>

</LyncSettings>

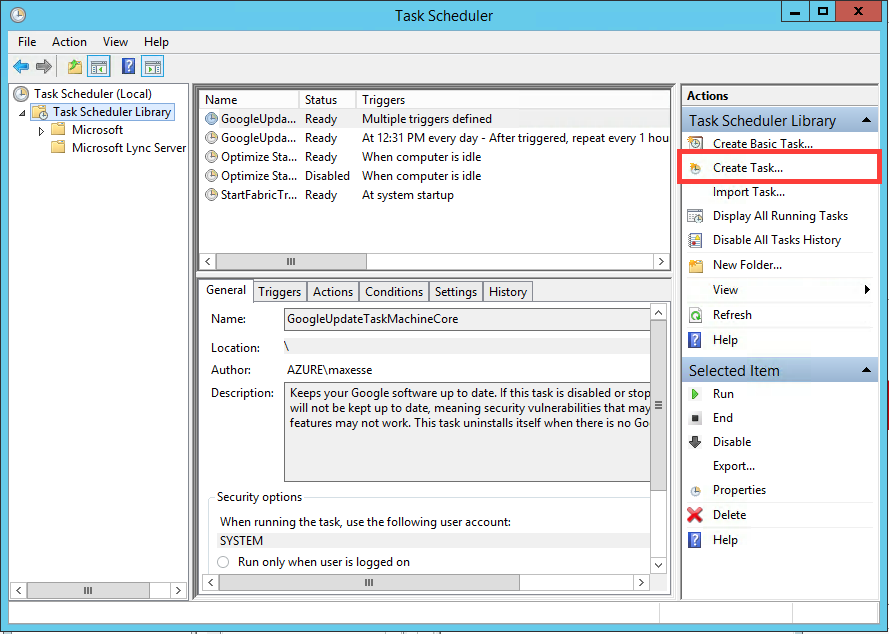
## Running LyncAutoOps unattended

Normally, once set up you’ll want this script to run without having to constantly babysit it, and that’s how it was designed in first place.

The easiest (and most normal way) to run something in a Windows environment is to take advantage of Windows Server’s task scheduler.

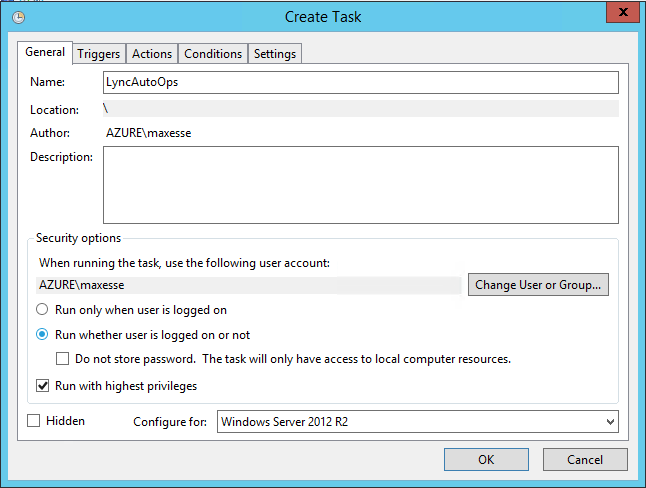
We’ll see here how to correctly create the task and make it run twice a day, every 12 hours.

First thing, start Task Scheduler and create a new task:

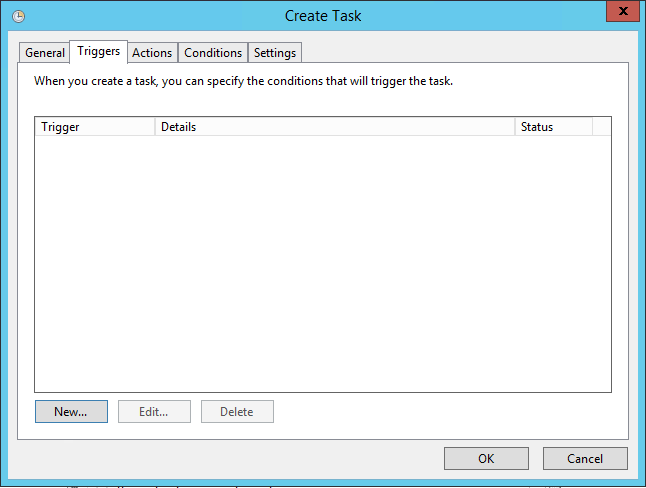
[](http://blog.maxsanna.com/wp-content/uploads/2015/01/Screen-Shot-2015-01-05-at-15.13.06.png)

Next, in the General Tab, set the following parameters:

* Name (obviously)
* Choose the service account that will run the script
* Tick ‘Run whether user is logged on or not’
* Tick ‘Run with highest privileges’ (it may or may not be required)
* Configure for: Windows Server 2012 R2 (or whatever highest level you’ve got available according to the version of Windows Server you’re running)

[](http://blog.maxsanna.com/wp-content/uploads/2015/01/Screen-Shot-2015-01-05-at-15.15.41.png)

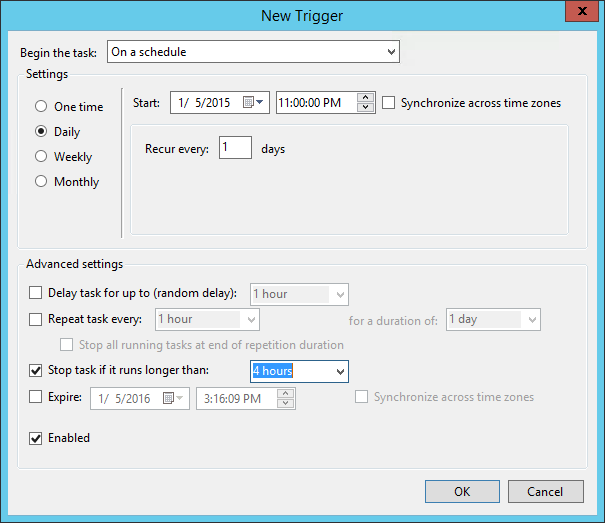
Next up, in the Triggers Tab, select the ‘New…’ button:

[](http://blog.maxsanna.com/wp-content/uploads/2015/01/Screen-Shot-2015-01-05-at-15.16.04.png)

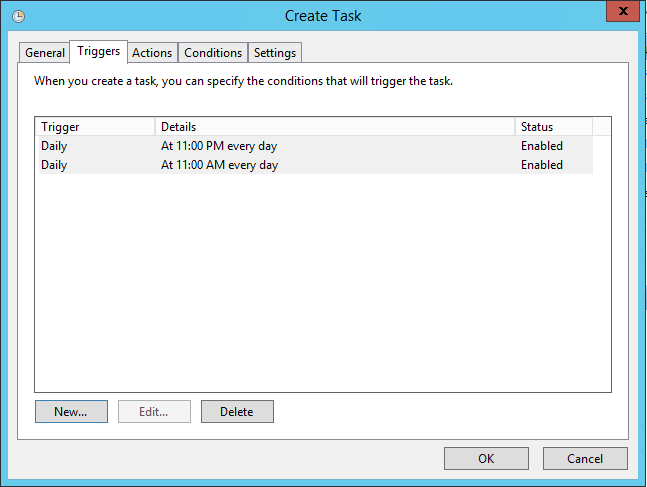
Configure the first trigger as follows:

* Run Daily
* Start it on the same day at 11:00:00 PM
* Recur every 1 days
* Stop task if it runs longer than 4 hours (you never know…)

Then add another task identical to the below, but set the start to the day after at 11AM (in this case that’d be 2/5/2015 at 11:00:00 AM)

[](http://blog.maxsanna.com/wp-content/uploads/2015/01/Screen-Shot-2015-01-05-at-15.16.38.png)

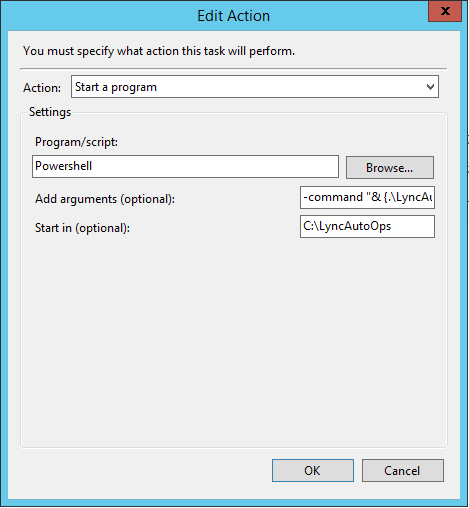
Here’s what the trigger window will look like once you’re done:

[](http://blog.maxsanna.com/wp-content/uploads/2015/01/Screen-Shot-2015-01-05-at-15.17.05.png)

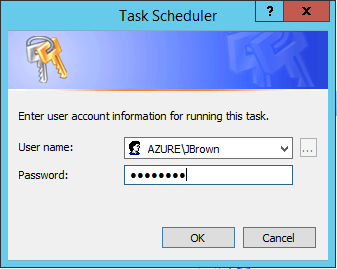
Next click on the Actions tab and create a new action. This step is very important. Set it as follows:

* Action: Start a program
* Program/script: Powershell
* Add arguments: -command “& {.\LyncAutoOps.ps1}”
* Start in: C:\LyncAutoOps

The ‘start in’ folder is very important, as all the config files and logs will be created relatively to this folder.

[](http://blog.maxsanna.com/wp-content/uploads/2015/01/Screen-Shot-2015-01-05-at-15.23.36.png)

Now you can confirm and save the entire task. The Task Scheduler will prompt you for the service account password now, as it will need it to run your task.

[](http://blog.maxsanna.com/wp-content/uploads/2015/01/Screen-Shot-2015-01-05-at-15.19.22.png)

That’s it!

To test if the script works, the quickest way is to disable a test Lync user in AD, then click Run manually, and check the Logs folder, to see whether a new log file was created.