Server Administration Toolkit

- Last update
- Software Requirements
- Description
- Tools
- Standalone Executable
- Code
- Tools Explained
- Change Log

Last update: March 5, 2013

Software Requirements:

- ArcGIS for Server 10.1
- ArcGIS for Desktop 10.1 Basic and higher (for interactively running tools)
- Python 2.7.+ (installed with Desktop and Server)

Author: Kevin Hibma (khibma@esri.com)

Description

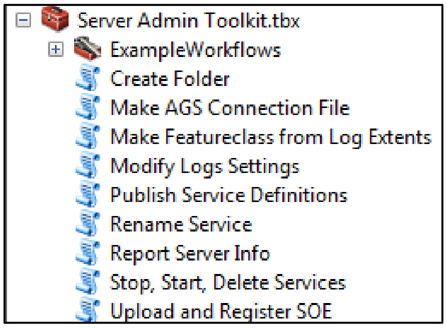
These tools perform some common administrative tasks with an ArcGIS Server machine. All of these tasks can be accomplished through the UI (ArcMap), the Web Manager or the REST Administration page. By using tools you can automate redundant workflows or chain common workflows together. Most of these tasks, turned into tools, have more detailed explanations in the help.

This package is composed of three main parts: **Tools, Standalone executable**, and **Code**.

Tools

The tools (described below) are found inside the *Tools* folder and can be used within ArcMap, or called from a python script. All tools have a dependency on the **arcpy** module and can only be run on a machine with ArcGIS Server, Desktop or Engine installed.

Move or copy the toolbox and all scripts to a working directory. Using ArcMap or ArcCatalog, navigate to the folder, expand the toolbox **Service Tools** and launch a tool by double clicking it. Provide input parameters and execute.

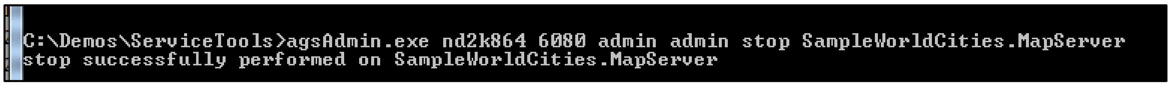


⊟ Server Admin Toolkit.tbx
- ⊞ ExampleWorkflows
- Create Folder
- Make AGS Connection File
- Make Featureclass from Log Extents
- Modify Logs Settings
- Publish Service Definitions
- Rename Service
- Report Server Info
- Stop, Start, Delete Services
- Upload and Register SOE

Standalone executable

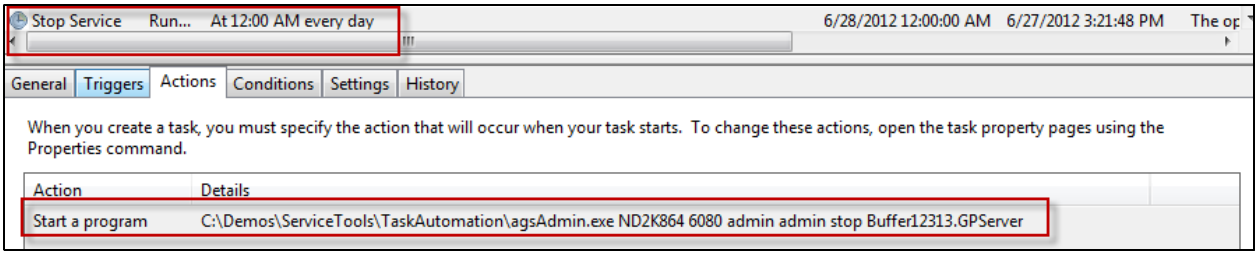
The standalone executable (agsAdmin.exe) found inside the *CommandLine* folder allows you to make command line calls to perform functions on your Server. Similar to the AGSSOM.exe utility you can place the .exe on a machine without any Esri or Python software and make calls for starting or stopping services. The executable can be used together with batch files (.bat) to create automated tasks. Or you can use Windows Task Sch

eduler to call the agsAdmin.exe and perform administrative tasks.



```
C:\Demos\ServiceTools>agsAdmin.exe nd2k864 6080 admin admin stop SampleWorldCities.MapServer
stop successfully performed on SampleWorldCities.MapServer
```

Eg 1: Using agsAdmin.exe at command line



Stop Service   Run...   At 12:00 AM every day   6/28/2012 12:00:00 AM   6/27/2012 3:21:48 PM   The op

When you create a task, you must specify the action that will occur when your task starts. To change these actions, open the task property pages using the Properties command.

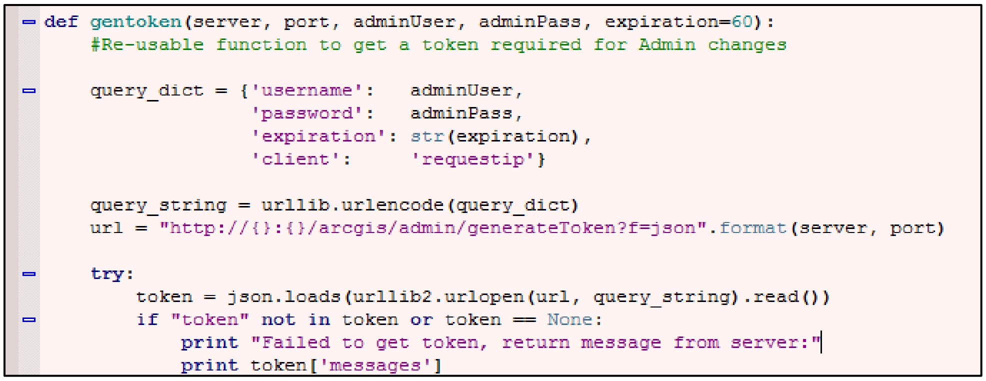
Action	Details
Start a program	C:\Demos\ServiceTools\TaskAutomation\agsAdmin.exe ND2K864 6080 admin admin stop Buffer12313.GPServer

Eg 2A scheduled task, set to stop a map service at midnight every day

Inside the TaskAutomation folder is enhanceMe.txt. This file provides instructions on adding your own functions and enhancements to the script and re-building the .exe.

Code

The code folder provides all the python code required to execute the same commands the tools do, without the requirement of the **arcpy** module (excluding the *Make AGS connection* and *Publish Service Definitions* tools as they have arcpy dependencies). The functions can be copied into your existing scripts or enhanced to perform specific tasks. For example, you could take the Stop/Start and getToken functions, put them into a new Python script and create a scheduled task to stop services automatically at a given time. Only basic error handling is done inside each function, you may want to extend error handling to create more robust scripts.



```
def gentoken(server, port, adminUser, adminPass, expiration=60):
    #Re-usable function to get a token required for Admin changes

    query_dict = {'username':   adminUser,
                  'password':   adminPass,
                  'expiration': str(expiration),
                  'client':     'requestip'}

    query_string = urllib.urlencode(query_dict)
    url = "http://{}:{}/arcgis/admin/generateToken?f=json".format(server, port)

    try:
        token = json.loads(urllib2.urlopen(url, query_string).read())
        if "token" not in token or token == None:
            print "Failed to get token, return message from server:"
            print token['messages']
```

Eg 3 the generate token function code, a self-contained function you can copy/paste

The code used inside the tools to pre-populate service lists and directories is provided in the code folder. You can use the tool validator code in the creation of your own script tools.

TOOLS

NOTE: Most tools require an administrative user and password to connect to the ArcGIS Server's REST Admin page. The user name and password is provided to the tool dialog using clear text and sent in clear text.

Create Folder

Creates a new folder on the ArcGIS Server

- Inputs: server, port, admin user, admin password, folder name, *folder description*

Make AGS Connection File

Creates a connection file for ArcGIS Server. A connection file (.ags) is required when publishing Service Definitions (.SD)

- Inputs: connection type, output connection location, connection name, server url, server type, *user name, password, save username*

Make Featureclass from Log Extents

Creates a feature class by parsing logs for a particular map service. The Server must have the log level set to INFO or greater for extent information to be saved.

- Inputs: server, port, admin user, admin password, map service, output featureclass

Modify Log Settings

Modifies the log level of an ArcGIS Server. Can also be used to clear existing logs.

- Inputs: server, port, admin user, admin password, clear logs?, log level

Publish Service Definitions

Publishes one or more Service Definitions (.SD) to an ArcGIS Server. The tool will auto populate available SDs from the input directory. If an optional server REST url is provided, existing folders are available for selection.

- Inputs: arcgis server connection type (.ags), folder of .sd, list of .sd to publish, start service immediately, *server rest url, folder*

Rename Service

Renames an existing service

- Inputs: server, port, admin user, admin password, service to rename, new name

Report Server Info

Prints out information about an ArcGIS Server to the messages of the tool execution.

- Inputs: server, port, admin user, admin password

Stop, Start, Delete Services

Provides the ability to stop, start or delete one or more services.

- Inputs: server, port, admin user, admin password, stop/start/delete, service(s)

Upload and Register SOE

Uploads a SOE (server object extension) to the Server and then registers it. Note, this tool has a dependency on the Python module Requests (included in this package)

- Inputs: server, port, admin user, admin password, soe (file)

Example Workflows toolset

Example workflows using tools to perform common tasks.

Clear Logs Quickly

- Edit the model and provide default values for the server connection information. You can clear the logs by launching the tool and executing.

Surface from Extents

- Create a hot spot map based on the tile requests for a map service. The centroid's of the extents are used and based on a 2 mile distance considered coincident.
- Make sure your dataframe is in the same coordinate system as the map service
- Note: The model requires the Spatial Analyst extension to perform Kernel Density. You could experiment with other tools such as Hot Spots to perform your analysis.

Make Connection and Publish SDs

- Chain two tools together to make an AGS connection and publish SDs from a given directory

Stop Services on a Server

- Edit the model and hardcode the server connection information as well as the option to STOP. When you open the tool you only need to select which services you want to immediately stop.

Change Log

March 5, 2013 – v1.2 – bug fixes:

- Rename tool now correctly renames services that exist in folders
    - Fixed both *renameService.py* and *code/AllFunctions.py*
- PublishSD tool now correctly honors if a service is to be published to a folder (that is: SD was created with folder information inside and no folder name was supplied to the tool)

July 11, 2012 – v1.0.1 – minor bug fixes:

- Python error when trying to act against service names with localized characters. For example, *Geológico*
    - Not all localized cases are supported, but toolkit provides *better* support than before

June 27, 2012 – v1.0 – initial release