



Esri International Developer Summit
Palm Springs, CA

ArcGIS for Server Administration API for C# Developers

Philip Heede

@pheede

Content

- **What is the REST admin API all about?**
- **Why script it- and why script it using C#?**
- **How to do it!**

What is the ArcGIS for Server REST Admin API

- RESTful interface for administering ArcGIS for Server
- Similar to the regular REST API used by the client SDKs and APIs
- Used by Manager and ArcGIS for Desktop when publishing or managing ArcGIS for Server

Why not just use Manager?

- Automating repetitive tasks
- Consistency in deployment
- Performing tasks not easily possible in Manager
- Automated monitoring
- The sky is the limit..

Why use C#?

- You could use any scripting or programming language capable of making HTTP GET and POST requests
- C# is widely used by many organizations and may be best fit for existing processes, install scripts, etc.
- Lots of other options: Python, Javascript, Java, Ruby, ..

Setup

- **Visual Studio 2010 SP1, Visual Studio 2012, or Visual Studio 2013**
- **Install NuGet (or update VS2012/VS2013 built-in to latest)**
- **In your C# project:**
 - **Install-Package Newtonsoft.Json**
- **I'm using System.Net.Http from .NET Framework 4.5**

My software for the presentation

- **Visual Studio 2013**
- **NuGet**
- **Json.NET (aka. Newtonsoft.Json)**
- **Fiddler**

The API doc

- **ArcGIS Server Administrator API for 10.2 documentation:**
<http://esriurl.com/agsadmin102>
- **One stop shop for resources, operations, and samples including JSON spec.**

First things first: get a token

- At ArcGIS for Server 10.1-10.2 all administrative requests must be authenticated with an administrative token
- `/arcgis/admin/generateToken` operation
 - Username
 - Password
 - Client
 - Encrypted
 - Other parameters depending on Client value
 - Pro tip: when `encrypted=true` then *all* parameters other than *f* have to be encrypted!

Securely sending sensitive data.. like your password

BEST SECURITY COMES FROM USING SSL ON SERVER WITH HIGH STRENGTH CERTIFICATE!

- /arcgis/admin/publicKey resource exposes 512-bit RSA key



- RSA encryption details not (yet) in the documentation:
 - PKCS#1 v1.5 padding
 - UTF8 byte encoding of strings
 - Hex encoding of bytes

Sending an HTTP request

- **.NET Framework 4.5 introduced System.Net.Http.HttpClient**
 - **Package from Microsoft adds HttpClient support to .NET Framework 4.0, Silverlight 4, Silverlight 5, Windows Phone 7.5, and Windows Phone 8:**
<http://nuget.org/packages/Microsoft.Net.Http>
- **Much simpler interface than classic HttpWebRequest or WebClient**
- **If working on older platform just use a different way to put together data and request**

Putting together HTTP request data

- Identify whether resource or operation is designed for GET or POST

```
URL:          http://server:port/arcgis/admin/services/{<folder>}/report
HTTP Method:  GET
```

```
URL:          http://server:port/arcgis/admin/generateToken
HTTP Method:  POST
```

- For GET requests parameters go in the query string
- For POST requests the data goes in the HTTP body

Encoding query string values for GET requests

- `System.Web.HttpUtility.ParseQueryString`
- Useful class for ensuring proper encoding of special characters and doing string manipulation for you.
- One query string parameter you'll always add: `f=json`

Creating HTTP body for POST requests

- `System.Net.Http.FormUrlEncodedContent`
- Part of `HttpClient` helper classes. Takes care of encoding POST data in the format used by the ArcGIS Admin API.

(exception: uploads/upload operation uses Multipart format)

- `IEnumerable<KeyValuePair<string, string>> data = ...;`
- `var content = new FormUrlEncodedContent(data);`
- Hint: `Dictionary<string, string>` is an `IEnumerable<KeyValuePair<string, string>>`

Parsing data: dealing with JSON

- **Json.NET (aka. Newtonsoft.Json) is widely used in the .NET community**
- **Open source (MIT license)**
- **Fast and flexible**

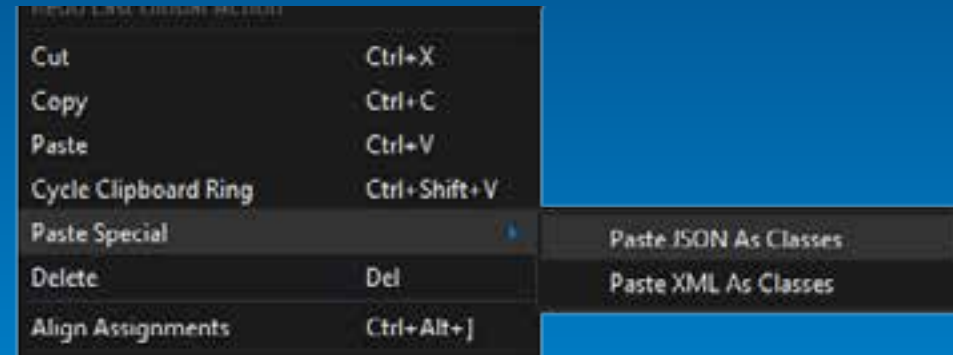


Option 1: DOM-style parsing

- `JObject result =
 Newtonsoft.Json.Linq.JObject.Parse(str);`
- Full LINQ capabilities as well as simply reading individual properties.
- `string attributeValue =
 result["attributeName"].Value<string>()`
- Great for one-off development.

Option 2: serializing to/from classes

- Create POCO classes: plain .NET classes that map to the JSON
- Visual Studio 2012 Update 2 and Visual Studio 2013 simplify task immensely with *Paste JSON As Classes* option



- Touch up generated classes with enums, number types etc.

Encoding conventions

- Raw byte arrays are hex encoded (used for encrypted data)
- Timestamps are represented as *milliseconds* since Unix epoch (1970-01-01).

Steps to interacting with REST Admin API

- 1. Get an administrative token
- 2. Identify input parameters and request type
- 3. Create data and send request
- 4. Parse response
 - Identify error conditions if any

Building a working sample

- **Goal: create a WPF watchdog application that shows service status (running or stopped)**
- **Steps:**
 - **Authenticate**
 - **Get list of folders**
 - **Get list of services and their status in each folder**
- **Demo time!**

Building a working sample

- **Goal: publish a service based on a pre-created .SD file**
- **Steps:**
 - **Authenticate**
 - **Upload .SD file to server**
 - **Create Service based on uploaded file and service properties**
- **Demo time!**

Getting the samples

- All code for the samples is available on GitHub:
- <https://github.com/pheede/agsadmin-devsummit>

Questions?



Understanding our world.