Agenda

- Introduction
- Server Object Extensions
  - Demo
- Server Object Interceptors
  - Demo
- Q&A
Introduction
ArcGIS Enterprise

- Powerful built-in mapping and analytics
- Configurable tools and templates
- Extensible APIs and building blocks
Why?
Motivations for extending ArcGIS Enterprise

- Existing behavior is not adequate
- Custom business logic
- Security
- Integrating disparate systems
Options

- Custom applications
- Geoprocessing services
- Server object extensions
- Server object interceptors
Custom applications

Business logic coded in your application

- Performance
- Reusability
- Scalability
Geoprocessing services

- Full GIS library
- Asynchronous
- Python

“Bread, butter for a GIS analyst”
Server Object Extensions and Interceptors

- Allows you to enhance capabilities of Map, Feature and Image services
- Services have to be published from ArcMap
- Extensions and Interceptors are not yet supported on services published from ArcGIS Pro**
- Must have ArcObjects development experience
- Java or .NET
Server Object
Extensions
Server Object Extensions
Basics

• Server-side code
• Attaches to Map or Image services
• **Custom API**
  - Needs custom applications
• Managed lifecycle
  - No additional hardware needed to deploy
  - No need to worry about authentication and authorization
  - One instance of your object per instance of the service
Use cases

- Editing
- Data Processing
- Image Functions
- Network Analysis
Architecture

Out-of-box applications

Well-defined API

Image Services

Map Services

Custom applications

Custom API

GIS Server

SOE

SOE
Development process

1. Install SDK
2. Start with samples/templates
3. Add logic and build .soe file
4. Register via Manager
5. Configure extension on a service
6. Debug
## Implementation notes

<table>
<thead>
<tr>
<th>Name</th>
<th>Functions</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IServerObjectExtension</td>
<td>init(...)</td>
<td>Initialization of SOE Cleanup</td>
</tr>
<tr>
<td></td>
<td>shutdown()</td>
<td></td>
</tr>
<tr>
<td>IRESTRequestHandler</td>
<td>getSchema()</td>
<td>Define the API as resources and operations</td>
</tr>
<tr>
<td></td>
<td>handleRESTRequest(...)</td>
<td>Handle all REST resources and operation requests</td>
</tr>
<tr>
<td>IRequestHandler2</td>
<td>handleStringRequest(...)</td>
<td>Handle SOAP requests</td>
</tr>
<tr>
<td></td>
<td>handleBinaryRequest(...)</td>
<td>Handle &quot;binary&quot; requests from ArcMap</td>
</tr>
<tr>
<td>IWebRequestHandler</td>
<td>handleStringWebRequest(...)</td>
<td>Handle OGC requests</td>
</tr>
</tbody>
</table>
Demo
Server Object Extensions
Server Object Interceptors
Server Object Interceptors

Basics

- Server-side code
- Filters requests to Map or Image services
- **No change in API**
  - Existing applications are unaware
- Two or more can form an ordered chain
- Managed lifecycle
  - No additional hardware needed to deploy
  - No need to worry about authentication and authorization
Use cases

- Security & Access Control
- Data Enrichment
- Notifications
- Auditing & Metering
Architecture

Out-of-box applications

Well-defined API

GIS Server

Map Services

Image Services

SOI
Call sequence

Out-of-box applications

GIS Server

ArcSOC

Web Handler

SOI 1

SOI 2

Map Services
## Implementation notes

<table>
<thead>
<tr>
<th>Name</th>
<th>Functions</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IServerObjectExtension</td>
<td>init(...) shutdown()</td>
<td>Initialization of SOE Cleanup</td>
</tr>
<tr>
<td>IRESTRequestHandler</td>
<td>getSchema()</td>
<td>Define the API as resources and operations</td>
</tr>
<tr>
<td></td>
<td>handleRESTRequest(...)</td>
<td>Handle all REST resources and operation requests</td>
</tr>
<tr>
<td>IRequestHandler2</td>
<td>handleStringRequest(...)</td>
<td>Handle SOAP requests</td>
</tr>
<tr>
<td></td>
<td>handleBinaryRequest(...)</td>
<td>Handle “binary” requests from ArcMap</td>
</tr>
<tr>
<td>WebRequestHandler</td>
<td>handleStringWebRequest(...)</td>
<td>Handle OGC requests</td>
</tr>
</tbody>
</table>
Demo
Server Object Interceptors
Road ahead
Coming to a release near you…

- New Enterprise SDK for Java and .NET developers
- Extend and intercept ArcGIS Pro-based Map, Feature and Image services
- Intercept calls to Utility Networks
Best practices and tips
Troubleshooting

• Logging
  - Access to server logger in your code
  - Messages can be logged at various levels (SEVERE, WARNING, INFO, …)
  - You can query logs using ArcGIS Server Manager or Administrator Directory (Admin API)
  - Always good to add logs in the initialization routines

• Make sure to package your dependencies!
• Be aware of service startup time (default: 5mins) if initialization takes longer
Debugging

• 
  Hint: Process command line contains the name of the service
  - Example: -Dservice=parcels.MapServer

• .NET
  - Attach Visual Studio to running ArcSOC process

• Java
  - Using ArcGIS Server Manager, enable remote debugging and configure port range
  - Configure IDE for remote debugging
  - Attach IDE to running ArcSOC process on a specific port
Always…

**Samples & templates**
Automate development workflows using Admin API

Use GP services for long running tasks
Don’t change the API in an SOI

**Implement security carefully**
No SOIs for hosted feature or tile requests
May need to increase ArcSOC heap size for intensive extensions
Questions?
How can we help you?