



ArcGIS GeoEvent Processor for Server: Extending with New Processors and Connectors



Ming Zhao
Software Developer
ArcGIS GeoEvent Processor for Server
mzhao@esri.com



Patrick Hill
Software Developer
ArcGIS for Military Solutions Team
patrick_hill@esri.com

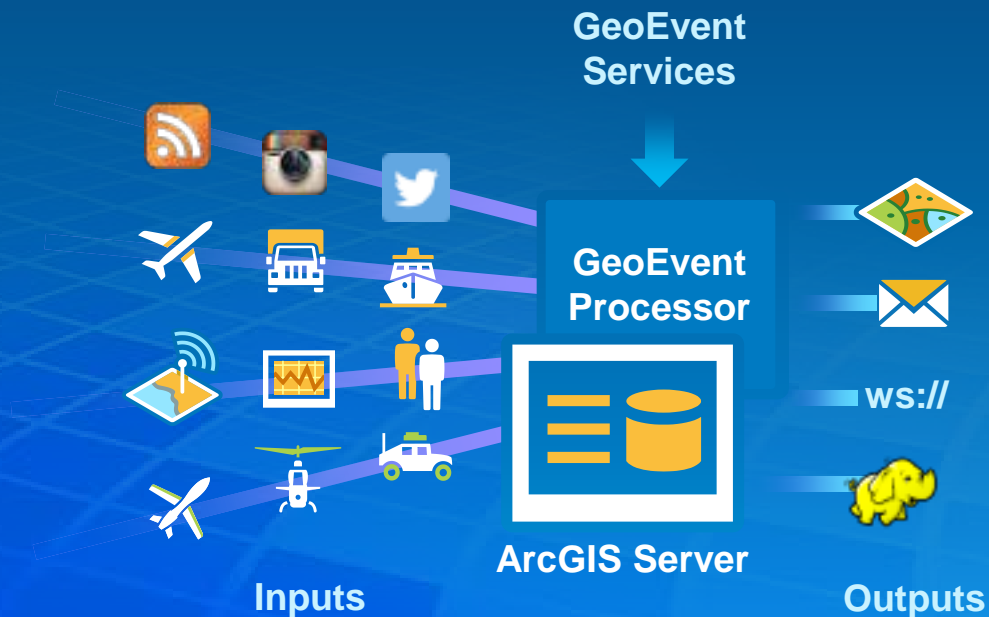
Technical Workshop



ArcGIS GeoEvent Processor for Server

Integrates and exploits real-time data

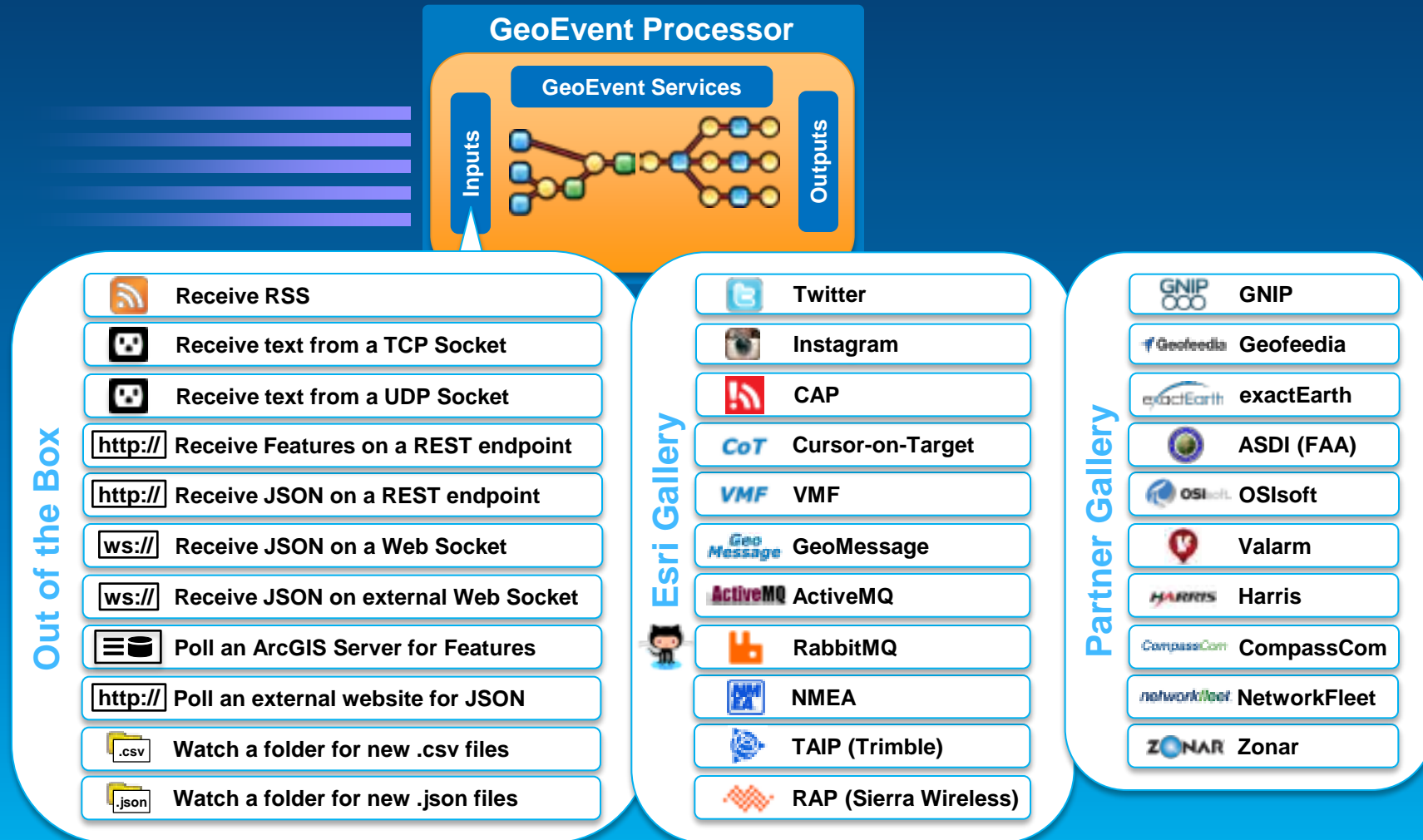
- Integrates real-time streaming data into ArcGIS
- Performs continuous processing and real-time analytics
- Sends updates and alerts to those who need it where they need it



Receiving real-time data

Connectors

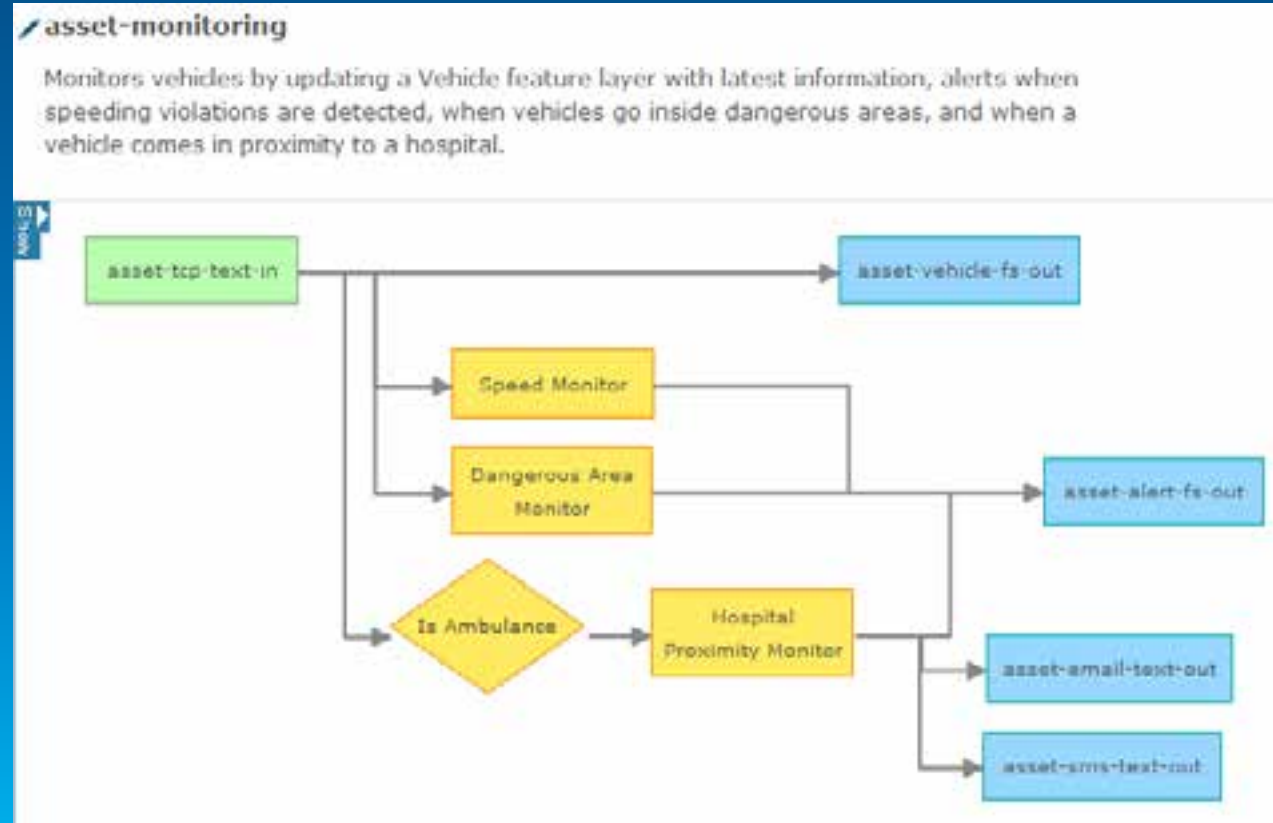
- You can easily integrate real-time data with ArcGIS by using an input **connector**.



Applying real-time analytics

GeoEvent Services

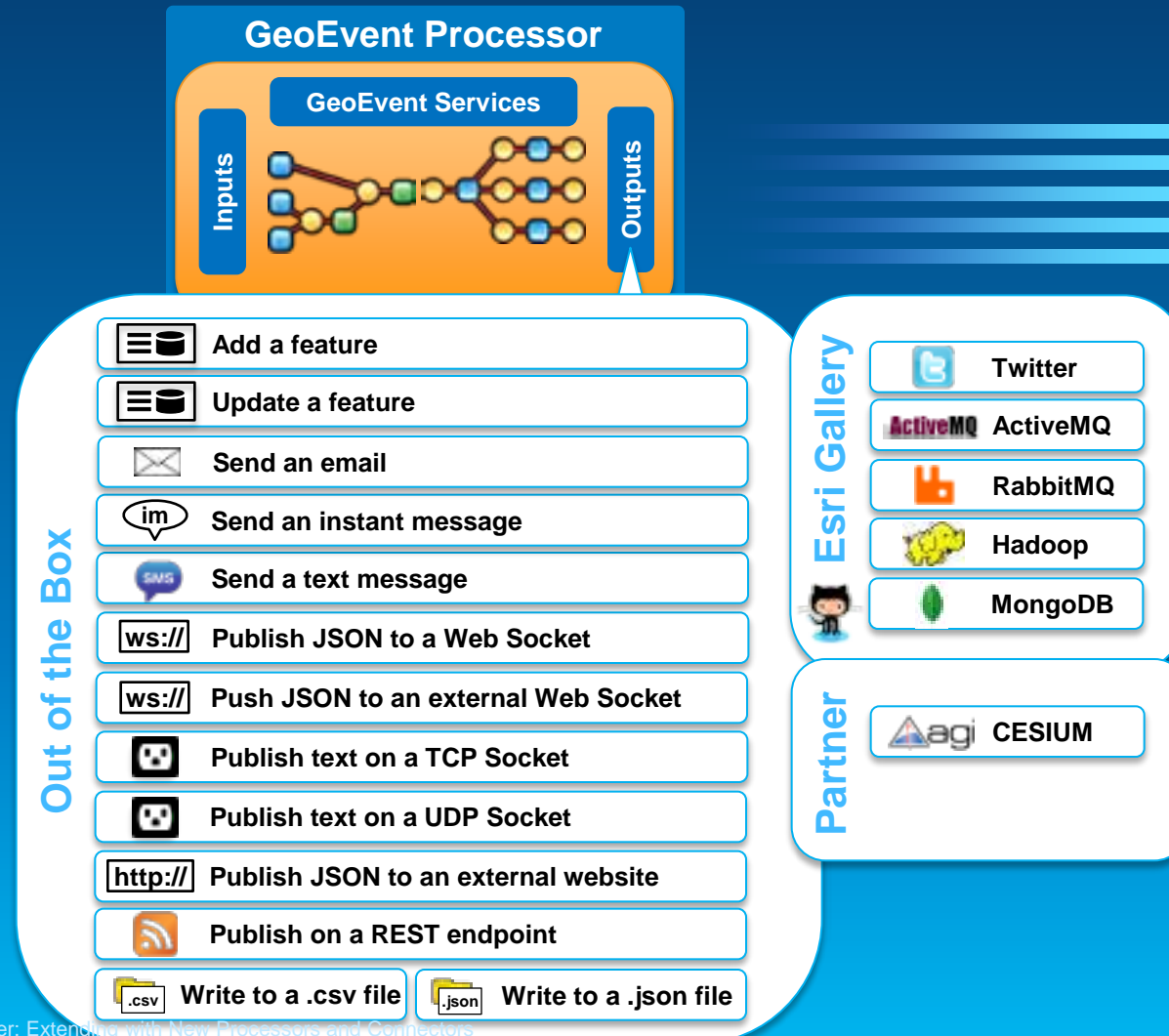
- A **GeoEvent Service** configures the flow of GeoEvents,
 - the **Filtering** and **GeoEvent Processing** steps to perform,
 - what input(s) to apply them to,
 - and what outputs(s) to send the results to.



Sending real-time data

Connectors

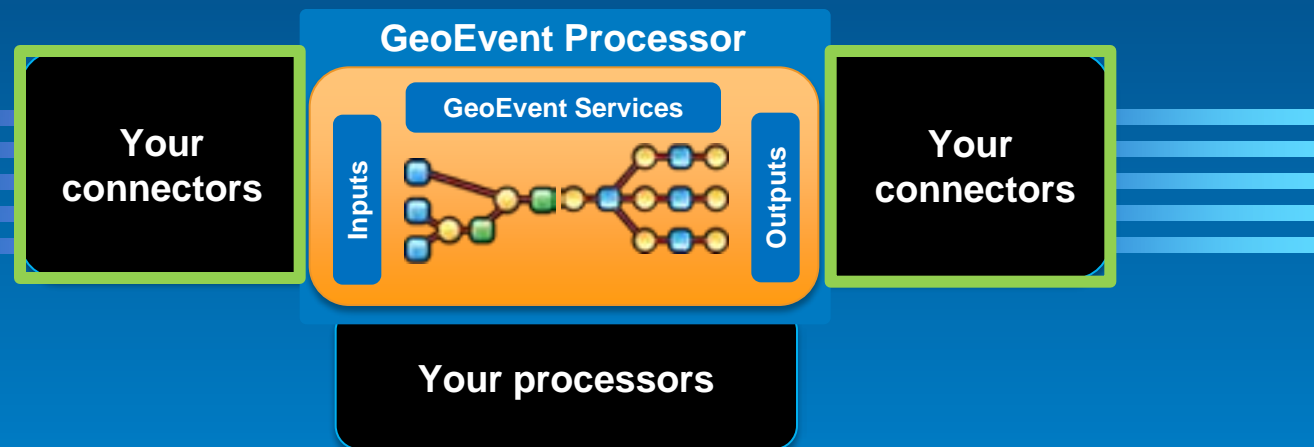
- You can easily send updates and results to those who need it where they need it using an output **connector**.



Extending GeoEvent Processor

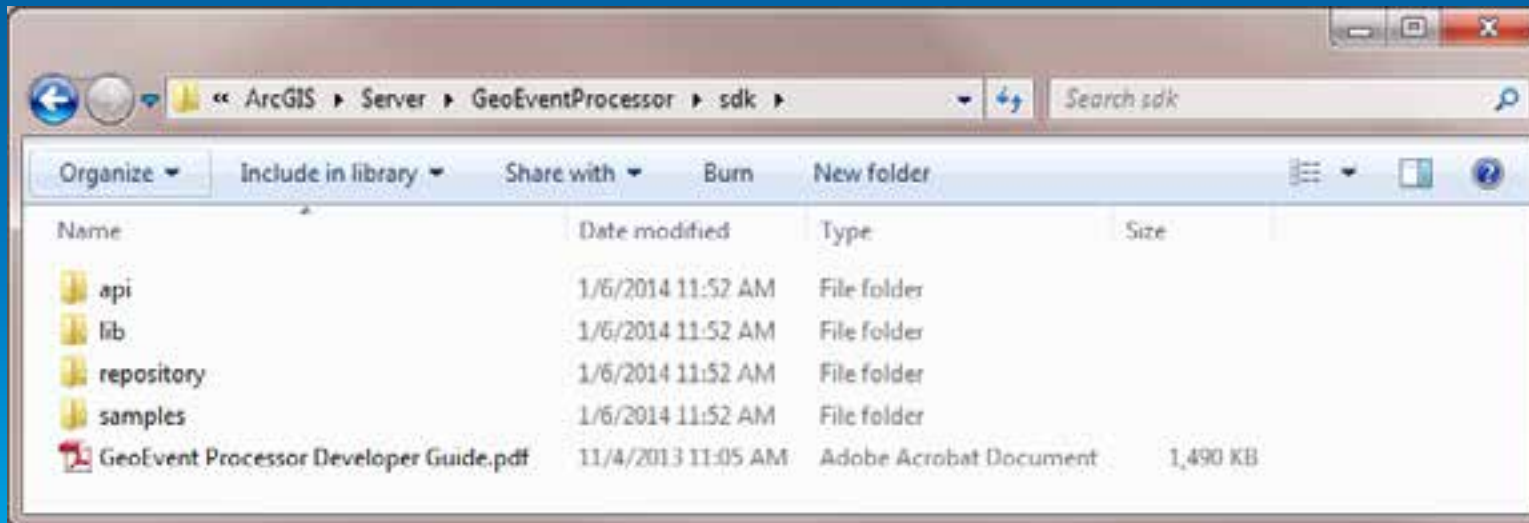
Software Development Kit (SDK)

- You can create your own custom connectors and processors using the GeoEvent Processor **Software Development Kit (SDK)**.



GeoEvent Processor SDK

- **api:** JavaDoc content associated with GeoEvent Processor SDK
- **lib:** Contains library used to build connectors (and processors)
- **repository:** Local maven repository
- **samples:** Sample connectors (and processors)
- **GeoEvent Processor Developer Guide**



Connectors



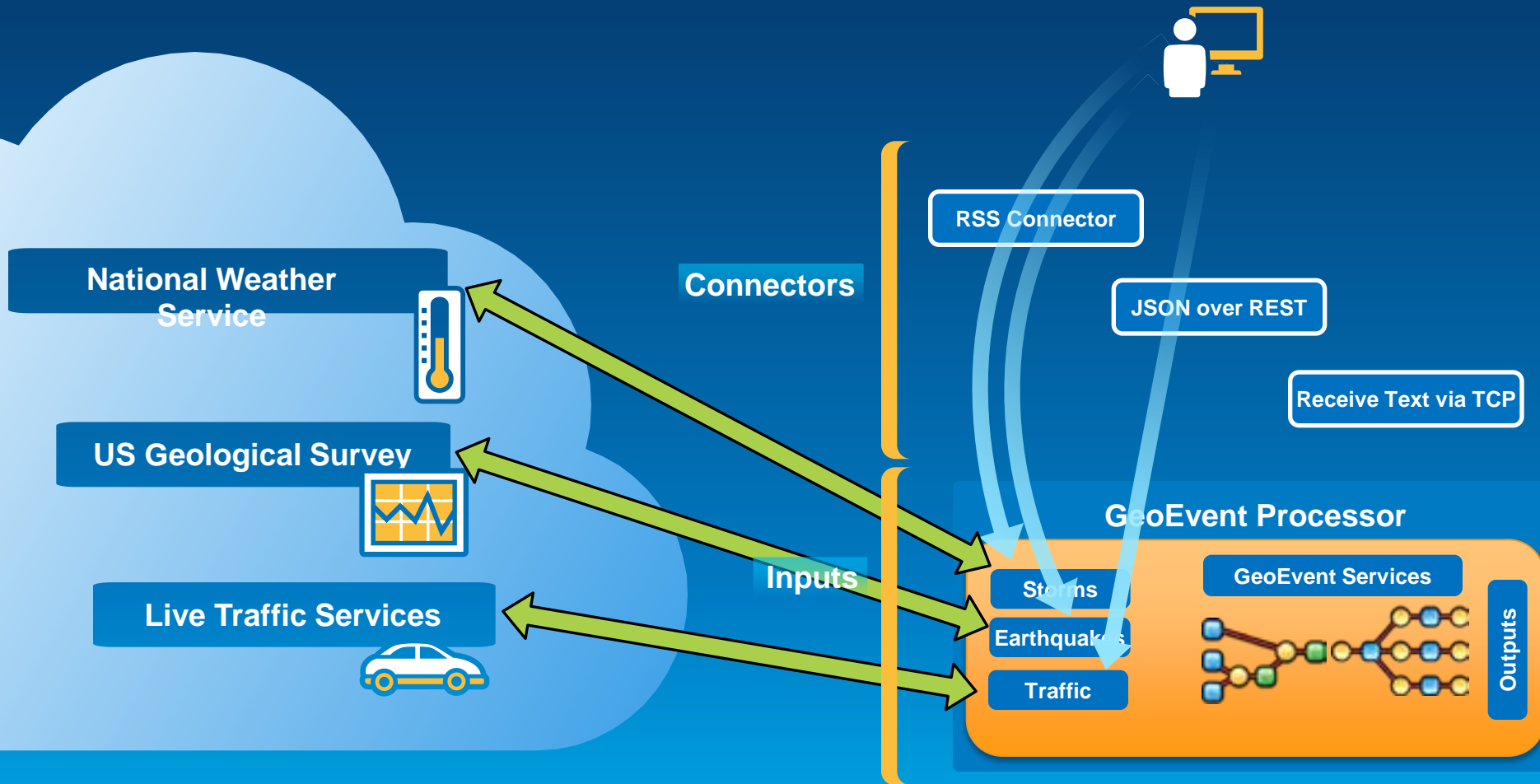
Connectors

What Does a Connector Do?

- **Connectors** are used to create **Inputs** and **Outputs**, hiding the technical details
- It might be very specific
 - Get latest earthquakes from USGS
- Or more general
 - Connect to an RSS feed

Creating Inputs

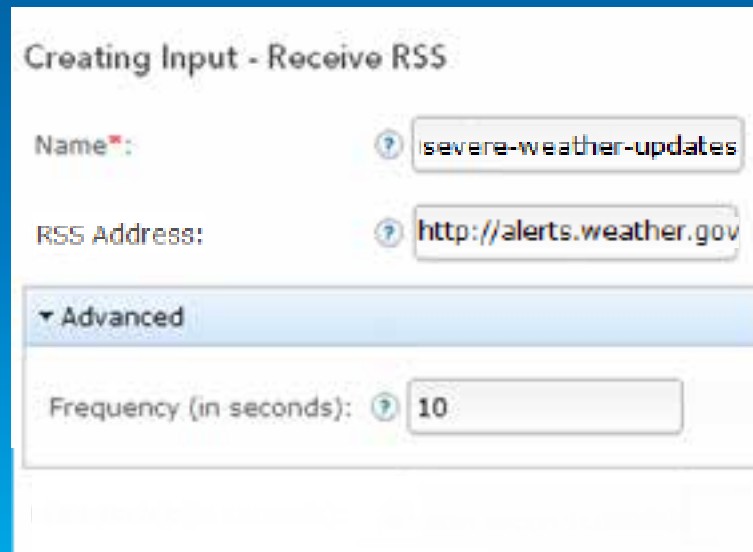
Configured using Connectors



Connector

How Does it Help

- The connector helps the user by
 - § Providing default values
 - § Re-label properties to be appropriate to the context
 - § Move properties under an “advanced” area to discourage modification
 - § Completely hide properties that the user should not see



Creating Input - Receive RSS

Name:

RSS Address:

▼ Advanced

Frequency (in seconds):

Connector

How Does it Help

- By choosing a **Connector**, the user implicitly selects **components** from the GeoEvent Processor that know
 - § HOW to move data (**Transport**)
 - § WHAT the data looks like (**Adapter**)

Example Input



Demo

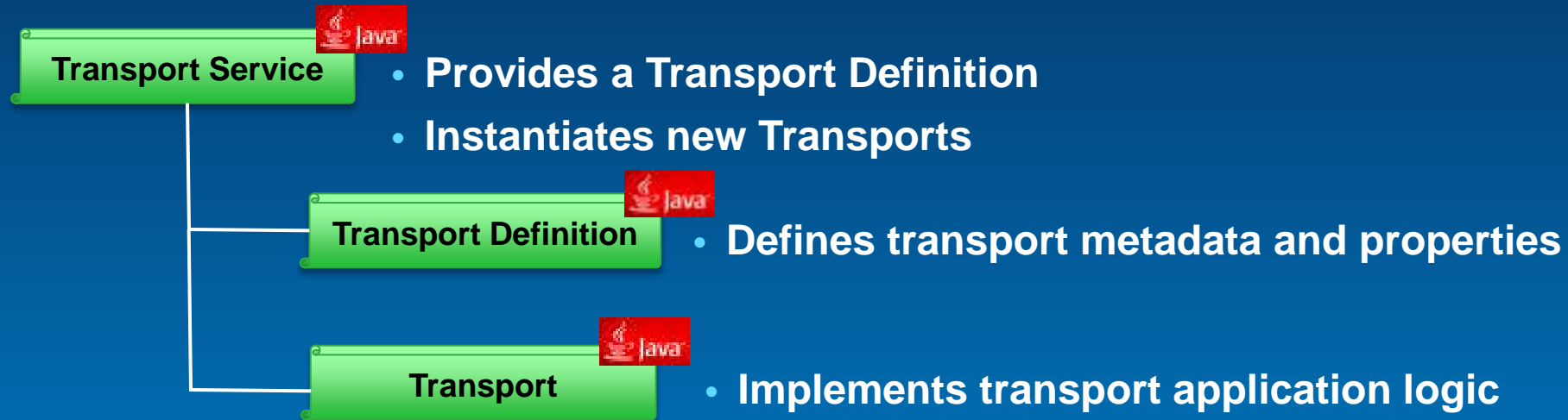
Exploring a Connector

Transport



Transport

What makes up a Transport?



Transport Behavior

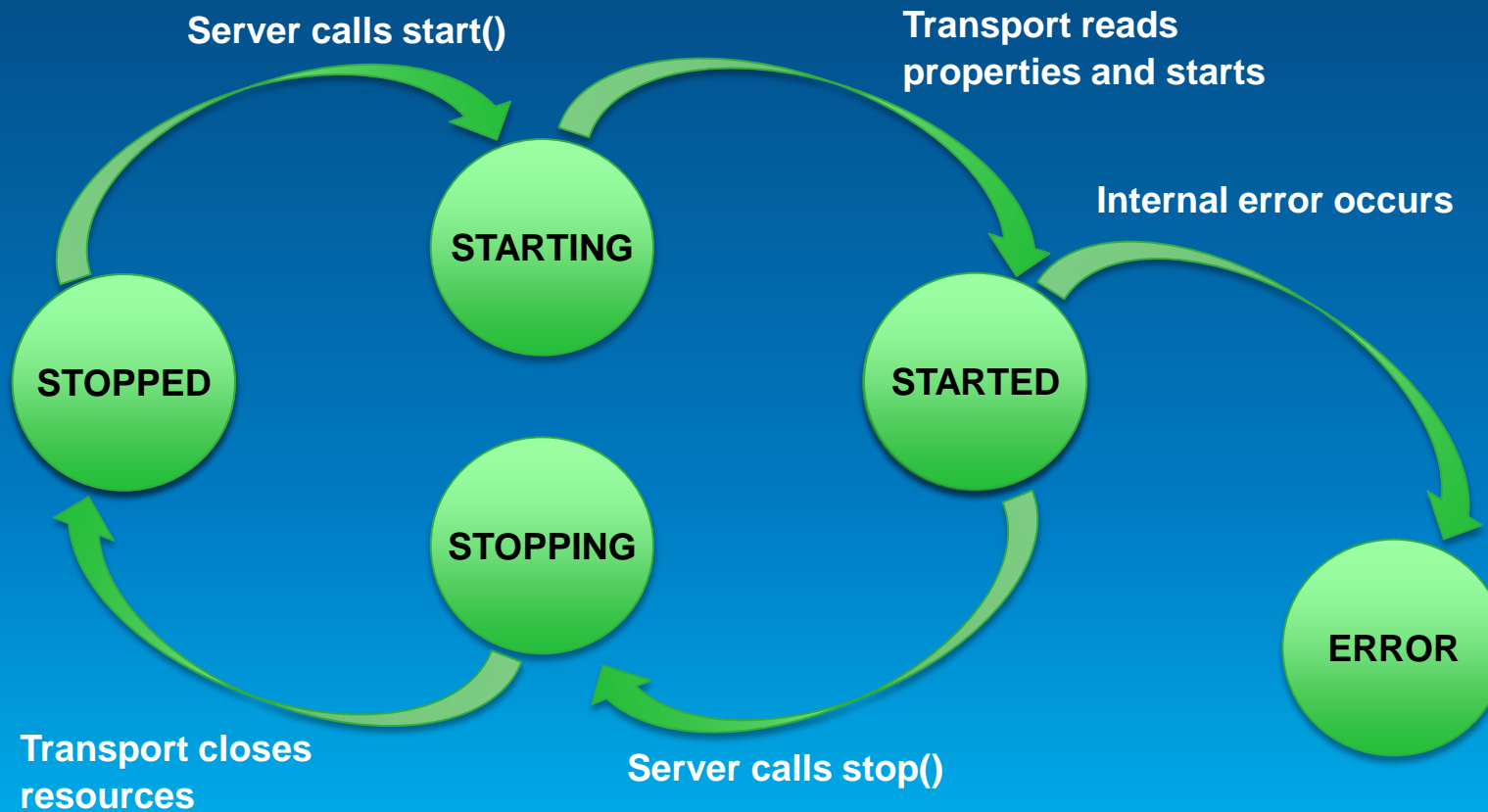
Transports

- **Transports are given**
 - § **Properties defining behavior**
 - § **A “ByteListener” where the bytes should be sent**
- **Transport is started by the server and it sends bytes to the receiver**
- **Transport is stopped by the server and it stops sending bytes**

Transport Lifecycle

Transports

Transports have a lifecycle that determines if they are producing data.



Outbound Transports

Transports

- **Outbound Transports accept arrays of bytes from the Adapter and transmit them.**
- **Occasionally the destination for the bytes depends on content in the GeoEvent.**
 - § **The Transport has the option of “looking back” at the GeoEvent that generated the bytes, and using it to route the data.**

Transport Code Walkthrough

- Starting the Transport

```
public class SampleInboundTransport extends InboundTransportBase implements Runnable
{
    private static final Log log = LogFactory.getLog(SampleInboundTransport.class);
    private static final int BUFFER_SIZE = 2048;
    protected DatagramSocket socket = null;
    private ByteBuffer byteBuffer = ByteBuffer.allocate(BUFFER_SIZE);
    private Thread thread = null;

    public SampleInboundTransport(TransportDefinition definition) throws ComponentException
    {
        super(definition);
    }

    @SuppressWarnings("incomplete-switch")
    public void start() throws RuntimeException
    {
        switch (getRunningState())
        {
            case STARTING:
            case STARTED:
            case STOPPING:
                return;
        }
        setRunningState(RunningState.STARTING);
        thread = new Thread(this);
        thread.start();
    }
}
```

Transport Code Walkthrough

- Receiving Data

```
public void run()
{
    try
    {
        applyProperties();
        setRunningState(RunningState.STARTED);
        byte[] buffer = new byte[BUFFER_SIZE];
        while (getRunningState() == RunningState.STARTED)
        {
            try
            {
                DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
                socket.receive(packet);
                String channelID = packet.getAddress().toString().replace('/', ' ').trim();
                byteBuffer.put(buffer, 0, packet.getLength());
                byteBuffer.flip();
                byteListener.receive(byteBuffer, channelID);
                byteBuffer.compact();
            }
            catch (SocketTimeoutException ste)
            {
                ....
            }
            catch (BufferOverflowException boe)
            {
                ....
            }
            catch (Exception e)
            {
                ....
            }
            socket.close();
            setRunningState(RunningState.STOPPED);
        }
    }
    catch (Throwable ex)
    {
        {
            log.error(ex);
            setRunningState(RunningState.ERROR);
        }
    }
}
```

Transport Code Walkthrough

- Applying Properties
- Stopping Transport

```
public void applyProperties() throws Exception
{
    int port = 1010;
    if (getProperty("port").isValid())
    {
        int value = (Integer) getProperty("port").getValue();
        if (value > 0 && value != port)
        {
            port = value;
        }
    }

    socket = new DatagramSocket(port);
    socket.setSoTimeout(100);
}

@SuppressWarnings("incomplete-switch")
public synchronized void stop()
{
    switch (getRunningState())
    {
        case STARTING:
        case STARTED:
            setRunningState(RunningState.STOPPING);
            break;
        case ERROR:
            setRunningState(RunningState.STOPPED);
            break;
    }
}
```

Properties

ArcGIS GeoEvent Processor Manager Services

Monitor **Inputs** GeoEvent Services Outputs

Creating Input - Poll an external website for JSON

Name*:

URL:

Create GeoEvent Definition*: Yes No

GeoEvent Definition Name (New):

HTTP Method:

▼ Advanced

Receive New Data Only: Yes No

Frequency (in seconds):

Header Parameter Name:Value List:

Post From:

Post Body:

Post body MIME Type:

Construct Geometry From Fields*: Yes No

JSON Object Name:

Parameters:

Use URL Proxy: Yes No

Expected Date Format:

Property Definition

Properties

- Transports and Adapters request properties through their “Definition” class.
- Each requested property has a
 - § Name
 - § Description
 - § Type (String, Integer, Float, ...)
 - § Default Value

```
<transport name="SampleInboundTransport" label="Sample Inbound Transport"
  contact="geoeventprocessor@esri.com" version="10.2.1" domain="sample.transport.inbound"
  type="inbound">
  <description>This is a sample inbound transport.</description>
  <propertyDefinitions>
    <propertyDefinition propertyName="port" label="Port"
      description="This is the UDP port." propertyType="Integer"
      defaultValue="1010" mandatory="true" readOnly="false" />
  </propertyDefinitions>
</transport>
```

Advanced Topics – More on Properties

Properties

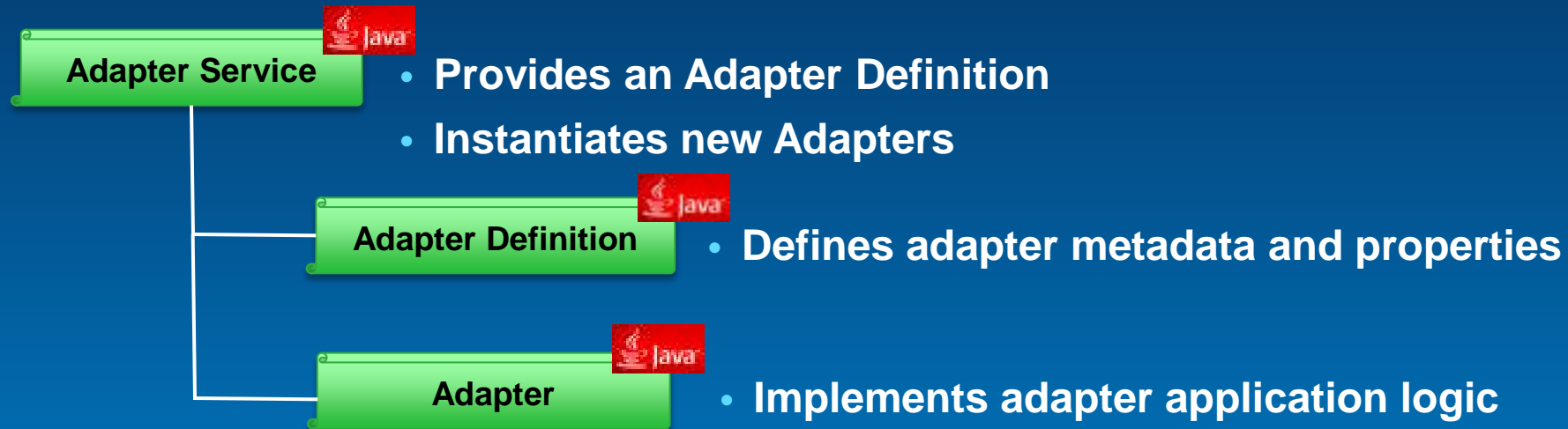
- **Properties can be Mandatory or Optional.**
- **Some properties are dependent on others**
 - § **Example: “Compression Algorithm” depends on “Compressed = True”**
- **Some properties only accept values from a list of Allowed Values**
 - § **“Compression Algorithm” : [Run Length Encoding, Zip, LZW]**

Adapter



Adapter

What makes up an Adapter?



Adapter Behavior

Adapters

- **Adapters are given**
 - § **Properties defining behavior**
 - § **A “GeoEventListener” where the GeoEvents should be sent**
- **Adapters are DATA DRIVEN**
 - § **No start/stop calls**
 - § **The adapter is handed a byte array and pushes any generated GeoEvents to the Listener**

Adapter Code Walkthrough

- Adapter Properties

```
public class SampleInboundAdapterDefinition extends AdapterDefinitionBase
{
    public SampleInboundAdapterDefinition()
    {
        super(AdapterType.INBOUND);
        try
        {
            GeotventDefinition md = new DefaultGeotventDefinition();
            md.setName("SampleGeoEventDefinition");
            List<FieldDefinition> fieldDefinitions = new ArrayList<FieldDefinition>();
            fieldDefinitions.add(new DefaultFieldDefinition("track id", FieldType.Integer));
            fieldDefinitions.add(new DefaultFieldDefinition("location", FieldType.Geometry, "GEOMETRY"));
            md.setFieldDefinitions(fieldDefinitions);
            geotventDefinitions.put(md.getName(), md);
        }
        catch (ConfigurationException ex)
        {
            ;
        }
    }
}
```

Adapter Code Walkthrough

- Adapter Properties

```
@Override
public String getName()
{
    return "Sample";
}

@Override
public String getLabel()
{
    return "Sample Inbound Adapter";
}

@Override
public String getDomain()
{
    return "sample.adapter.inbound";
}
```

```
@Override
public String getDescription()
{
    return "This is a sample adapter.";
}

@Override
public String getContactInfo()
{
    return "yourname@yourcompany.com";
}

@Override
public String getVersion()
{
    return "10.2.1";
}
```

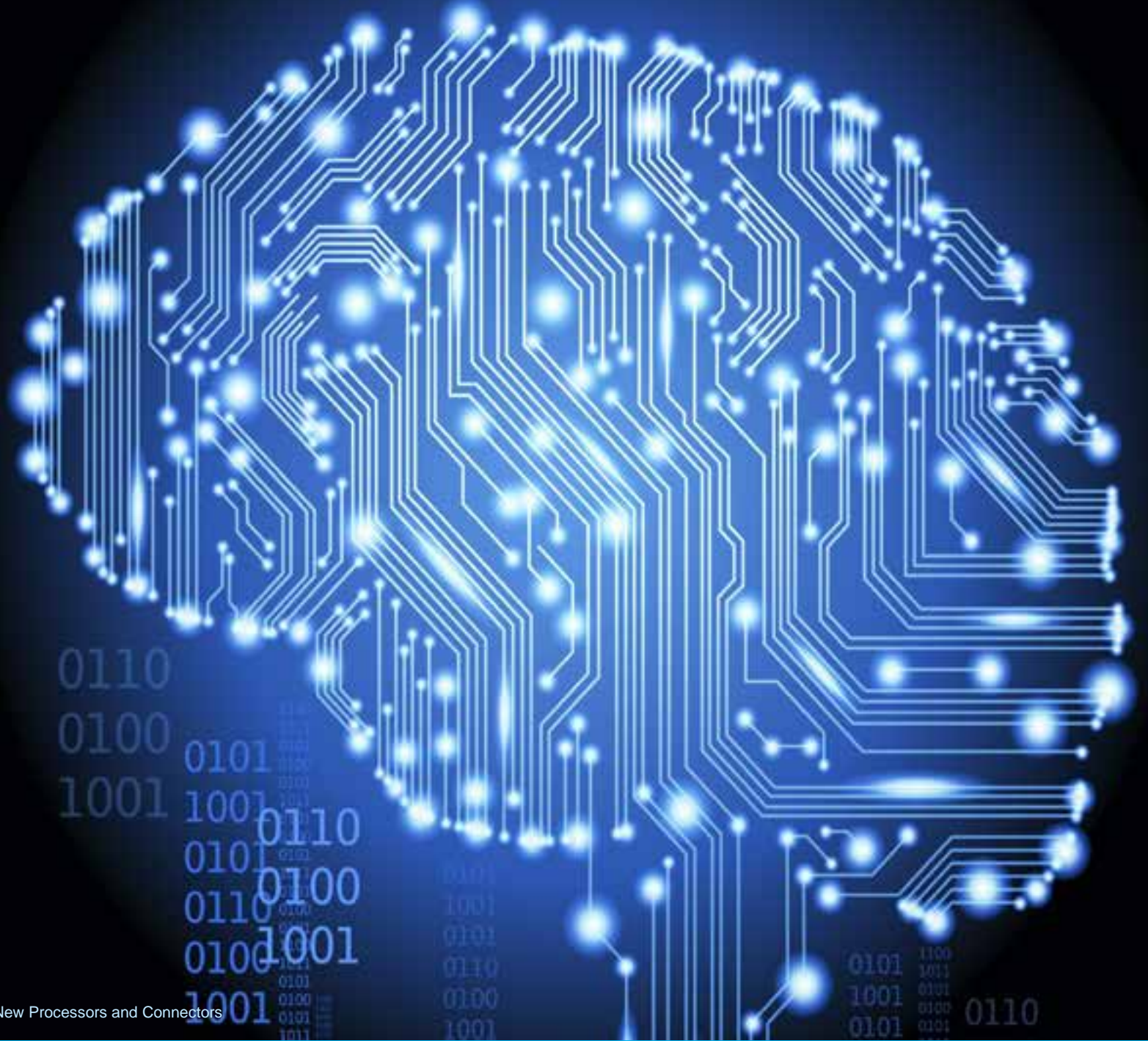
Adapter Code Walkthrough – Parsing Data

```
public GeoEvent adapt(ByteBuffer buffer, String channelId)
{
    buffer.mark();
    try
    {
        int id = buffer.getInt();
        double x = buffer.getDouble();
        double y = buffer.getDouble();
        GeoEvent msg;
        msg = geoEventCreator.create(((AdapterDefinition) definition).getGeoEventDefinition("SampleGeoEventDefinition").getGuid());
        msg.setField("track_id", id);
        msg.setGeometry(spatial.createPoint(x, y, 4326));
        return msg;
    }
    catch (MessagingException e)
    {
        return null;
    }
    catch (BufferUnderflowException ex)
    {
        buffer.reset();
        return null;
    }
    catch (FieldException e)
    {
        e.printStackTrace();
        return null;
    }
}
```

Demo

Installing and Using Custom Adapter and Transport

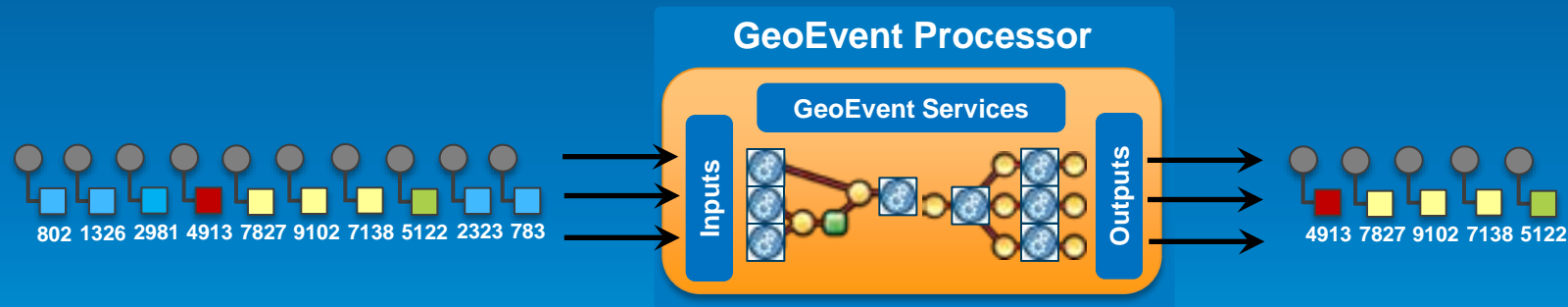
Processors



Extending GeoEvent Processor

What is a Processor?

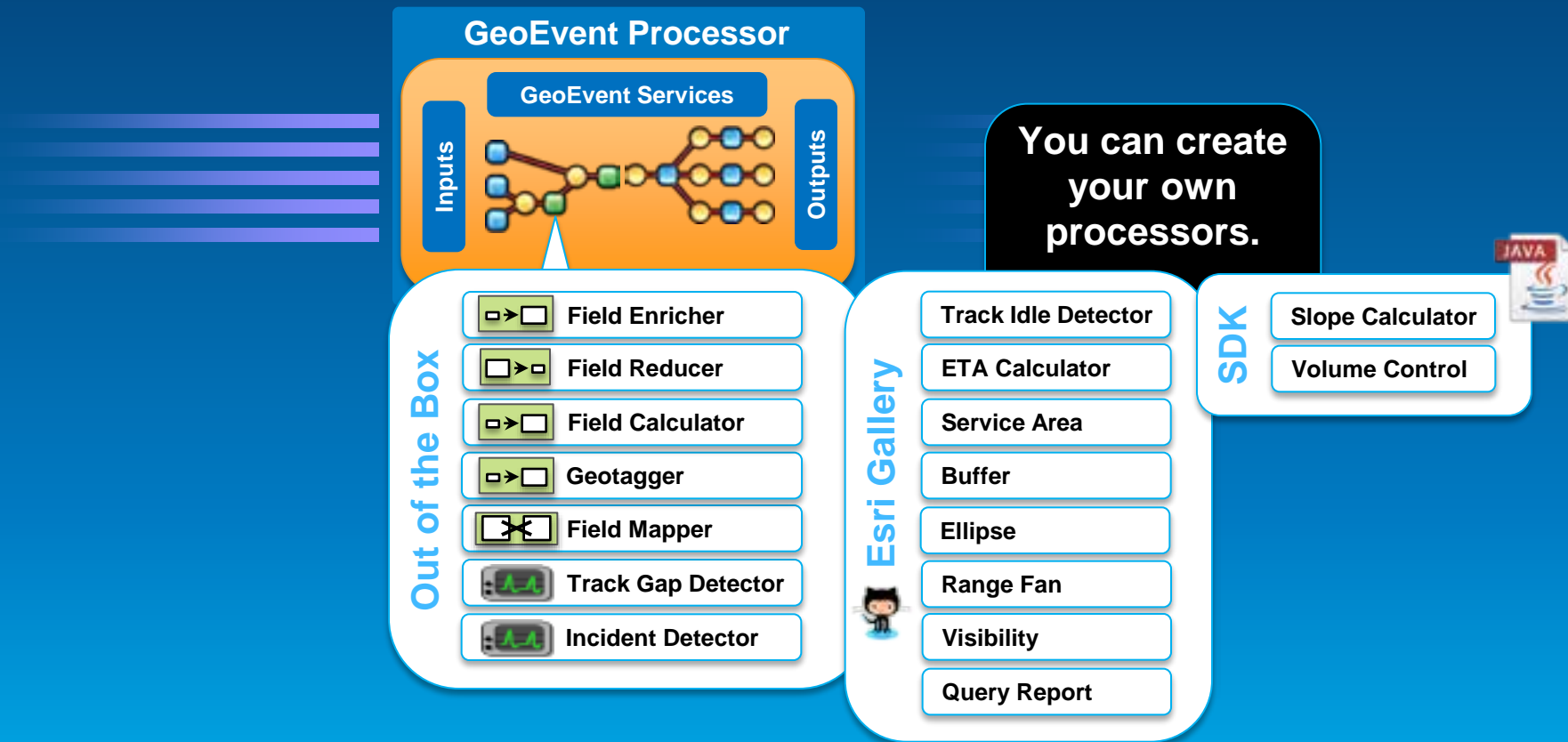
- Software component written in Java 
- User configurable
- Performs specific actions on GeoEvents
- Runs continuously inside of server processing environment
- Shares lifecycle with GeoEvent Service



Extending GeoEvent Processor

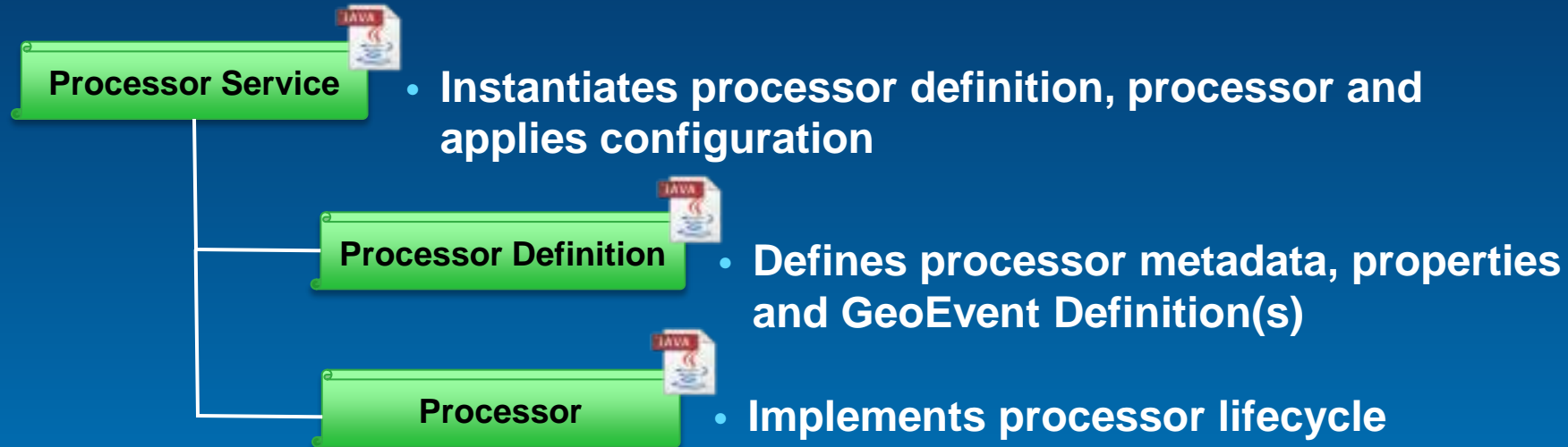
GeoEvent Processors

- You can perform continuous analytics on GeoEvents as they are received using a **processor**.



Anatomy of a Processor

What makes up a Processor?



Processor Definition

Step 1: Properties and GeoEvent Definition(s)

```
public class SampleProcessorDefinition extends GeoEventProcessorDefinitionBase
{
    private static Log LOG = LogFactory.getLog(SampleProcessorDefinition.class);

    public SampleProcessorDefinition()
    {
        try
        {
            // Define processor properties ...
            propertyDefinitions.put("propertyName", new PropertyDefinition("propertyName",
                PropertyType.String, "", "Label", "Description", false, false));

            // Define processor GeoEventDefinition(s) ...
            GeoEventDefinition ged = new DefaultGeoEventDefinition();
            ged.setName("SampleDefinition");
            List<FieldDefinition> fds = new ArrayList<FieldDefinition>();
            fds.add(new DefaultFieldDefinition("fieldName", FieldType.String, "FIELD_TAG"));
            ged.setFieldDefinitions(fds);
            geoEventDefinitions.put(ged.getName(), ged);
        }
        catch (Exception e)
        {
            LOG.error("Error setting up SampleProcessor Definition.", e);
        }
    }
}
```

Processor Definition

Step 1: Metadata

Identification

1. Name
2. Domain
3. Version

```
@Override
public String getName()
{
    return "SampleProcessor";
}

@Override
public String getDomain()
{
    return "sample.processor";
}

@Override
public String getVersion()
{
    return "10.2.2";
}
```

Description

4. Label
5. Description
6. Contact Information

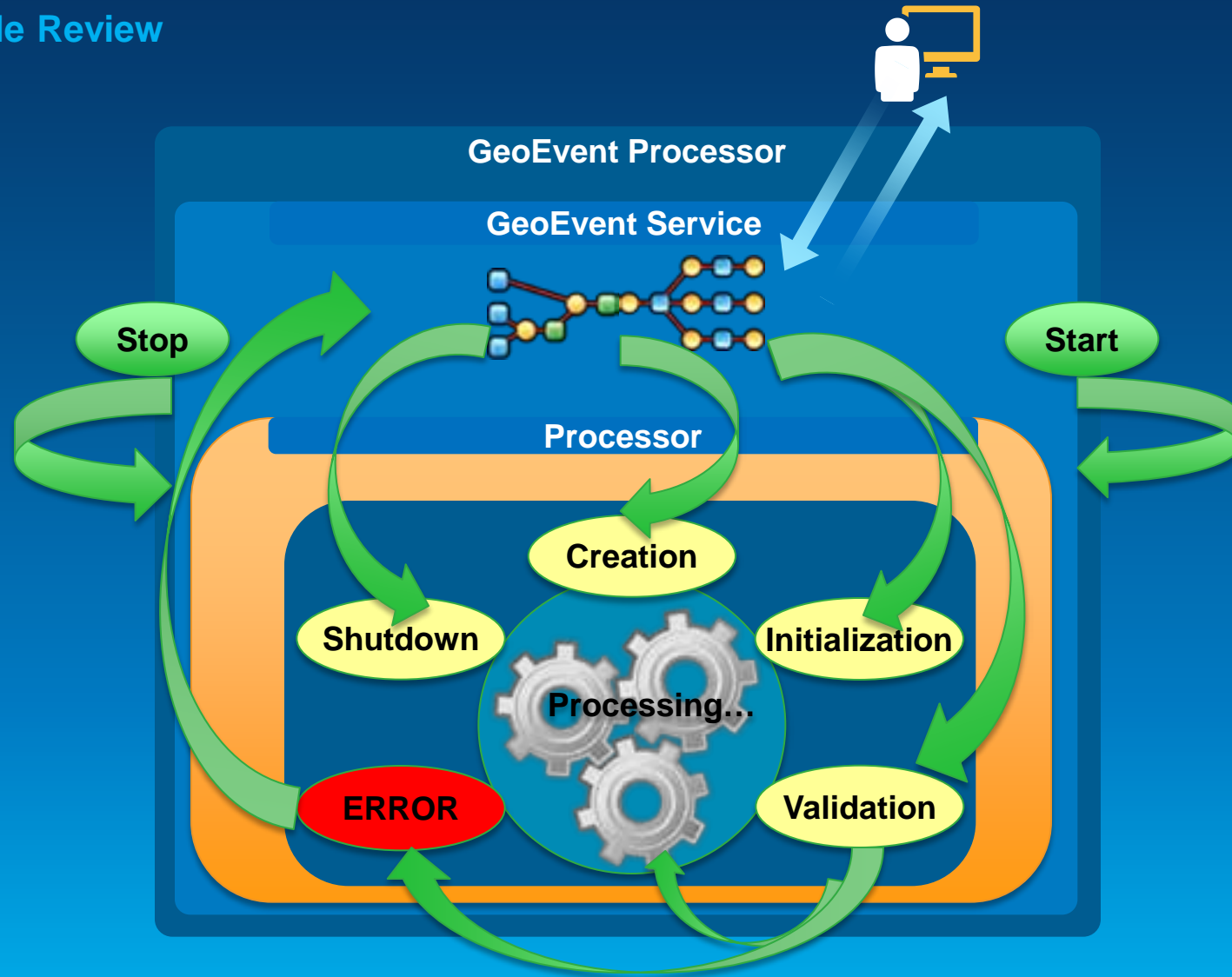
```
@Override
public String getLabel()
{
    return "Sample Processor";
}

@Override
public String getDescription()
{
    return "This is a Sample Processor.";
}

@Override
public String getContactInfo()
{
    return "geoeventprocessor@esri.com";
}
```

Processor

Step 2: Lifecycle Review



Demo

Property Definitions & Spatial API

Rangefan
Calculator

Processor

Step 2: Lifecycle

1. Creation
2. Initialization
3. Validation

```
public class SampleProcessor extends GeoEventProcessorBase
{
    private static final Log LOG = LogFactory.getLog(SampleProcessor.class);

    public SampleProcessor(GeoEventProcessorDefinition definition) throws ComponentException
    {
        // Creation
        super(definition);
    }

    @Override
    public void afterPropertiesSet()
    {
        // Initialization
    }

    @Override
    public synchronized void validate() throws ValidationException
    {
        // Validation
        super.validate();
    }
}
```


Processor

Step 2: Lifecycle

4. Processing
5. Shutdown
6. Service Start / Stop

```
@Override
public GeoEvent process(GeoEvent geoEvent) throws Exception
{
    // Processing
    return geoEvent;
}

@Override
public void shutdown()
{
    // Shutdown
    super.shutdown();
}

@Override
public void onServiceStart()
{
    // Service Start
}

@Override
public void onServiceStop()
{
    // Service Stop
}
```

Processor Service

Step 3: Definition

- Create new Java class
- Extend `GeoEventProcessorServiceBase` class
- Creates processor definition
- Creates processor instance
- Applies configuration

```
public class SampleProcessorService extends GeoEventProcessorServiceBase
{
    public SampleProcessorService()
    {
        definition = new SampleProcessorDefinition();
    }

    @Override
    public GeoEventProcessor create() throws ComponentException
    {
        return new SampleProcessor(definition);
    }
}
```

Processor Service

Step 4: Configuration

- Declare processor service as a service implementing `GeoEventProcessorService` interface from GeoEvent Processor SDK

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
  <bean id="sampleProcessorServiceBean"
        class="sample.processor.SampleProcessorService" activation="eager">
    <property name="bundleContext" ref="blueprintBundleContext" />
  </bean>
  <service id="sampleProcessorService" ref="sampleProcessorServiceBean"
    interface="com.esri.ges.processor.GeoEventProcessorService"/>
</blueprint>
```

Processor Service and Processor

Rangefan
Calculator



Demo

Putting it Together

Rangefan
Dashboard

Review

- **Connectors – Recipe for creating inputs/outputs**
 - **Transport – Moves raw data in/out of the GeoEvent Processor**
 - **Adapter – Converts raw data to GeoEvents and back**
 - **Properties – Used to configure an input/output for a specific use case**

GeoEvent Processor

Additional Resources

- **Developer Guide in the SDK**
- **Forum**
<http://forums.arcgis.com/forums/257-GeoEvent-Processor>
- **Resource Center – Includes Tutorials**
<http://pro.arcgis.com/share/goeevent-processor>
- **Browse the GitHub projects**

Thank you...

- **Please fill out the session survey:**

Offering ID: 1233

Online – www.esri.com/ucsessionsurveys

Paper – pick up and put in drop box

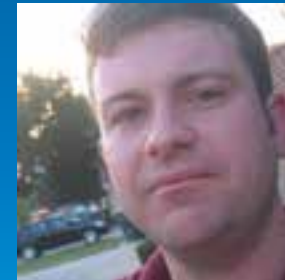
Questions / Feedback?

To learn more:

<http://pro.arcgis.com/share/geoevent-processor>



Ming Zhao | Software Developer
ArcGIS GeoEvent Processor for Server
mzhao@esri.com



Patrick Hill | Software Developer
ArcGIS for Military Solutions Team
patrick_hill@esri.com



Understanding our world.