Sharer code example

The following is a simple example of a **Sharer** handler for sharing messages to the Integration Mesh.

```
import com.perspectium.api.Message;
import com.perspectium.logging.PerspectiumLogger;
import com.perspectium.replicator.ASharer;
import com.perspectium.replicator.AgentConfig;

public class EchoSharer extends ASharer {
    final static PerspectiumLogger Log = PerspectiumLogger.create(EchoSharer.class);

    private static final String DEFAULT_ECHO_TEXT = "Hi there - this is an echoed message ";
    private static final String ECHO_MESSAGE = "echo_message";
    private static final String ECHO_ITERATIONS = "echo_iterations";

    private Message fMessage;
    private String fEchoText;
    private int fEchoIterations = 1;

    public void processMessages() {
        try {
            for (int iteration = 1; iteration <= fEchoIterations; iteration++) {
                String text = fEchoText + " " + String.valueOf(iteration);
                Log.info("publishing text: " + text);

                fMessage = new Message.Builder()
                        .topic(AgentConfig.getTopic(fTaskName))
                        .type(AgentConfig.getType(fTaskName))
                        .key(AgentConfig.getKey(fTaskName))
                        .name(AgentConfig.getName(fTaskName))
                        .value(text)
                        .build();

                publish(fMessage);
            }

            Log.info(String.format("task: %s echoed %s message(s) to the queue: %s",  fTaskName,
fEchoIterations, fQueueName));

        } catch (Exception e) {
            Log.error("Error:" + e.getMessage());
            e.printStackTrace();
        }
    }

    @Override
    public void initialize(final String taskName) {
        super.initialize(taskName);

        fEchoText = AgentConfig.getStringValue(fTaskName, ECHO_MESSAGE);

        if (fEchoText == null)
            fEchoText = DEFAULT_ECHO_TEXT;

        if (AgentConfig.taskHasElement(fTaskName, ECHO_ITERATIONS))
            fEchoIterations = AgentConfig.getIntValue(fTaskName, ECHO_ITERATIONS);

    }
}
```

As you can see, our EchoSharer class extends the abstract class ASharer which is required for all Sharer handlers. Your Sharer handler must implement the processMessages() method, which will be called by the framework when you're task is scheduled to run. More about scheduling will follow.

intialize

For now let's turn our attention to the initialize(String taskName) method. Your initialize method will be called with the taskName string set to the value of your <task_name> directive you defined within the conf/agent.xml configuration file. This is where you'll want to perform any configuration validation to ensure you have everything you need setup prior to your processMessages() method being called.

In this example, we accept two configuration directives being added within our <task> configuration. They are <echo_message> which accepts arbitrary text which will be the message we echo and <echo_iterations> which is an integer representing how many times we want to echo our message. For example:

```
<echo_message>This is a simple message that is being shared or echoed.</echo_message>
<echo_iterations>10</echo_iterations>
```

As you can see in the initialize method code example, you can use the AgentConfig.taskHasElement method to determine if your section of the overall configuration contains a given directive. You pass in your task name which acts an an index to your tasks configuration. Furthermore, you can use either the *AgentConfig.getIntValue(fTaskName, CONFIGURATION_DIRECTIVE)* or *AgentConfig.getStringValue(fTaskName, CONFIGURATION_DIRECTIVE)* to access the values of an integer or string-based type respectively.

We assume the <echo_message> will be provided and if it's not then we set a default message. Our approach for how many iterations is to assume it won't be configured so we establish a default and then override it if required. This example is contrived simply to show some available options.

Best practice is to throw a *ConfigurationException* if you can't establish a solid starting state within your initialize method. Otherwise your class should be ready for its processMessages method to be called.

processMessages

Our Sharer class simply shares a configured message or the default message a configured number of iterations or just once. What's important is the use of the publish method. The signature of the method is publish(Message message) where message is the Perspectium Message that will be published. The Perspectium Framework takes care of getting the message into the Perspectium Integration Mesh within the default outbound queue or the queue defined using the <message_connection> directive within your <task> configuration. The framework also handles encryption of the message prior to being placed into the Integration Mesh and reporting of message statistics such as its size.

You've successfully created a Sharer handler which will push your messages into the Integration Mesh for consumption by a Subscriber. Let's now write the Subscriber that will consume our echoed message(s).

For a **Subscriber** handler code example, see Subscriber code example.