# Reporting with the Agent SDK

The **Agent SDK** supports reporting errors and acknowledgements back to an Integration Mesh queue. This queue can then be subscribed to and consumed by the source that shared the messages (such as the Perspectium application in a ServiceNow instance or a custom Sharer handler) to be notified and/or handle any errors that the target consumer encountered.

## Procedure

To include reporting in your SDK application, follow these steps:

1

### Import the following package

```
import com.perspectium.replicator.reporting.Reporting;
```

2

### Instantiate a new Reporting object

```
Reporting reporter = new Reporting();
```

3

### Set batch size limit

Set the batch size limits in your agent.xml underneath your specified <task>. Once these limits are set the agent will send out a batch error or batch acknowledgement.

Default value: **1000**

```
<error_batch_size>42</error_batch_size>
<ack_batch_size>42</ack_batch_size>
```

## Sending Errors

You can send back custom errors to your sharing ServiceNow instance. For instructions on leveraging this method in your SDK application, please refer to the below java-doc.

```
/**
 * Report an error for the specified message.
 * @param error: The error string you would like to report.
 * @param message: The message that the error belongs to.
 */
public void reportError(String error, Message message) throws ReportingException;
```

## Handling Acknowledgements

Reporting also supports acknowledging messages through our existing Receipts functionality.

```
/**
* Process message and create acknowledgement to send back to ServiceNow sender.
* @param message: the message you are acknowledging.
*/
public void processAck(Message message) throws ReportingException;
```

## Reporting Example

Here is an example implementation that illustrates how to use both features.

```
package com.perspectium;

import com.perspectium.api.Message;
import com.perspectium.logging.PerspectiumLogger;
import com.perspectium.replicator.ASubscriber;
import com.perspectium.replicator.SubscribeException;
import com.perspectium.replicator.reporting.Reporting;
import com.perspectium.replicator.reporting.Reporting.ReportingException;

public class TestSubscriber extends ASubscriber {
        final static PerspectiumLogger Log = PerspectiumLogger.create(TestSubscriber.class);
        private final static String TASKNAME = "test_task";
        private final static String QUEUE = "psp.test.errors";
        final Reporting reporter = new Reporting(TASKNAME, QUEUE);

        public void postProcessMessage() {
                Log.info("POST Process TestSubscriber");
        }

        @Override
        public boolean processMessage(Message message) throws SubscribeException {
                if (message.getValue() == "") {
                        String errorMsg = "Message value empty!";
                        Log.error(errorMsg);
                        reportError(errorMsg, message);
                        return false;
                }
                processAck(message);
                return true;
        }

        private void reportError(String errorMessage, Message message) {
                try {
                        reporter.reportError(errorMessage, message);
                } catch (ReportingException e) {
                        // Reporting exceptions are thrown when reporting fails
                        // You can log it, try again, etc.
                        Log.error(e.toString());
                }
        }

        private void processAck(Message message) {
                try {
            Log.info("Sending acknowledgment back to ServiceNow");
                        reporter.processAck(message);
                } catch (ReportingException e) {
                        Log.error(e.toString());
                }
        }
}
```