Using the Agent SDK *Beta*

This tutorial is an overview of how to use the Agent SDK to build your own New Relic agent. You'll learn how to use the Agent SDK to instrument and measure the performance of your application's web transactions, and eventually more.

Note: This is pre-release software and should be treated as such. We are giving you early access to the beta to enable you to view web transactions. Please use this beta as an opportunity to provide feedback and guide us as we add more features and work towards a production release of the SDK.

A note before we get started

The purpose of the SDK is to allow you to monitor web applications that were not written in one of our supported languages. The instructions in this tutorial assume your application is written in C or C++. If you are not using C or C++, then you will need to create a wrapper around the APIs. The means of creating a wrapper will vary from language to language. For example, with perl you can use SWIG or XS.

Operating System and Compiler support

The libraries in the SDK were built using gcc 4.4.7 and gcc 4.6.3. If you want a release built using a different version of gcc, please create a feature request via the standard support channels.

The SDK only supports Linux at this time and has been tested with CentOS 6.3 and Ubuntu 12.04.

Choosing a usage mode

In order to maintain communication with New Relic, the Agent SDK requires a thread to be started that harvests data once per minute. This means that your app will need to be able to start a thread, which will run continuously as long as transactions are being executed. For some languages, this is simple to do. For others, there is no mechanism to start a thread within the same process running your web transactions.

For example, if you are running perl apps through Apache, then each perl script is a short-lived process that exits upon completion of the script. There is no ability to startup a thread that lives beyond the life of the script.

To solve this problem, we designed the SDK so that your agent can run in two different modes: daemon-mode or embedded-mode. In daemon-mode, you'll run the `newrelic-collector-client-daemon` that came with the Agent SDK as a stand-alone process. The daemon will collect and send data to New Relic. In embedded-mode your application will start a thread, which will have the same responsibilities as the daemon but will be

running within the same process as your transactions.

If you choose to run in daemon-mode read “How to run in Daemon-mode.” Otherwise read “How to run in Embedded-mode.” After that, you should read “How to Instrument Your Code.”

How to run in Daemon-mode

Security note: The SDK daemon will run with the privileges of the application starting it, which may be elevated. For example, if the daemon is started by Apache, it may run as root. You should run the daemon by a non-root user if possible.

Security note: The Agent SDK does not currently support SSL for communications between the SDK and New Relic. When naming web transactions, you should be careful not to include any sensitive information.

1. Set up your environment

Set the following environment variables before launching the daemon:

Environment variable	Format/Example
NEWRELIC_LICENSE_KEY	<insert your license key>
NEWRELIC_APP_NAME	My application
NEWRELIC_APP_LANGUAGE	perl
NEWRELIC_APP_LANGUAGE_VERSION	5.5
NEWRELIC_LOG_FILE_NAME	/path/to/log/file
http_proxy	http://<username>:<password>@<proxy_server>:<port>

2. Start the daemon process

Execute the following command in a terminal window:
`./newrelic-collector-client-daemon`

How to run in Embedded-mode

To run in embedded-mode, you'll need to use `libnewrelic-collector-client` in your web server process.

1. Embed `libnewrelic-collector-client` in your application

Follow these instructions to embed the collector client in your app:

1. Include the `newrelic_collector_client.h` and `newrelic_common.h` header files.

2. Set a logging level, e.g.:
   ```
   nr_setup_logging(NR_LOG_LEVEL_INFO, "/path/to/log/file");
   ```

3. Initialize the library.
   ```
   nr_init("my_license_key", "My Application", "perl", "5.5");
   ```

4. (Optional) Create a function to receive callback notifications when the library is shut down and register it:
   ```
   void my_shutdown_notice() {
        // do something
   }

   nr_register_shutdown_callback(my_shutdown_notice);
   ```

5. Register required callbacks to send data to New Relic when transactions are completed.
   ```
   /*
   By default, the transaction library expects an externally
   running daemon to handle web transactions. Since you are running
   in embedded-mode, you will need to register the
   nr_default_web_transaction_handler to process transaction data
   when transactions are completed.
   */
   nr_setup_embedded_collector_client(nr_default_web_transaction_ha
   ndler);
   ```

7. Link your application to `libnewrelic-common` and `libnewrelic-collector-client`.

How to instrument your code

Follow these instructions to measure transactions in your web app

1. Include the `newrelic_transaction.h` header file.
2. Surround the code being instrumented with the library's start and stop functions. After calling the start function, give your instrumented web transaction a name.
   ```
   /* Start web transaction */
   int transaction_id = nr_start_web_transaction();
   ```

```
/* Set name of transaction (e.g. show_user) */
/* See note below for naming recommendations */
nr_name_web_transaction(transaction_id, "show_user");

// Your code goes here

/* End web transaction */
int result = nr_end_web_transaction(transaction_id);
```

3. Link your application to `libnewrelic-common` and `libnewrelic-transaction`.

4. Use caution when naming web transactions. Issues come about when the granularity of web transaction names is too fine, resulting in hundreds if not thousands of different web transactions. If you name transactions poorly, your application may get blacklisted due to too many metrics being sent to New Relic.

   A basic rule is to name transactions based on the action rather than the URL. To learn more, read https://docs.newrelic.com/docs/features/metric-grouping-issues

That is all it takes to get up and running with the Agent SDK! Fire up your application and you will start seeing data in New Relic.