Error Detection

On this page:

You can customize how AppDynamics associates code events with business transaction errors by adding or removing the methods, log messages, HTTP codes, or redirect pages that are configured to indicate business transaction errors.

Permissions

To configure error detection, you need the **Configure Error Detection** permission.

Error Detection Configuration

You can view or modify error detection configuration in the **Error Detection** tab under **Configuration > Instrumentation**. Tabs take you to the configurable options for each programming language or platform. Note that some settings are platform specific, such as custom loggers or error detection using redirect pages, which are available for Java and .NET only.

To prevent a transaction from being marked as an error transactions in AppDynamics, clear the **Mark Business Transaction as error** check box. If clear, the transaction does not add to the error count metric and is considered in other metrics, such as response time, despite the occurrence of the error.

Supplement Error Detection

For Java and .NET, you can supplement the methods that AppDynamics considers error methods from common error handling frameworks with your own, custom-defined error methods by defining a customer logger. When the application invokes the method in a business transaction, AppDynamics considers the transaction to be an error transaction for metric purposes.

In the custom logger configuration, you can specify the parameter that contains the exception or error information passed to the method. This information will appear as the error message in AppDynamics, which invokes `toString()` on the parameter to extract the message.

To configure a custom logger

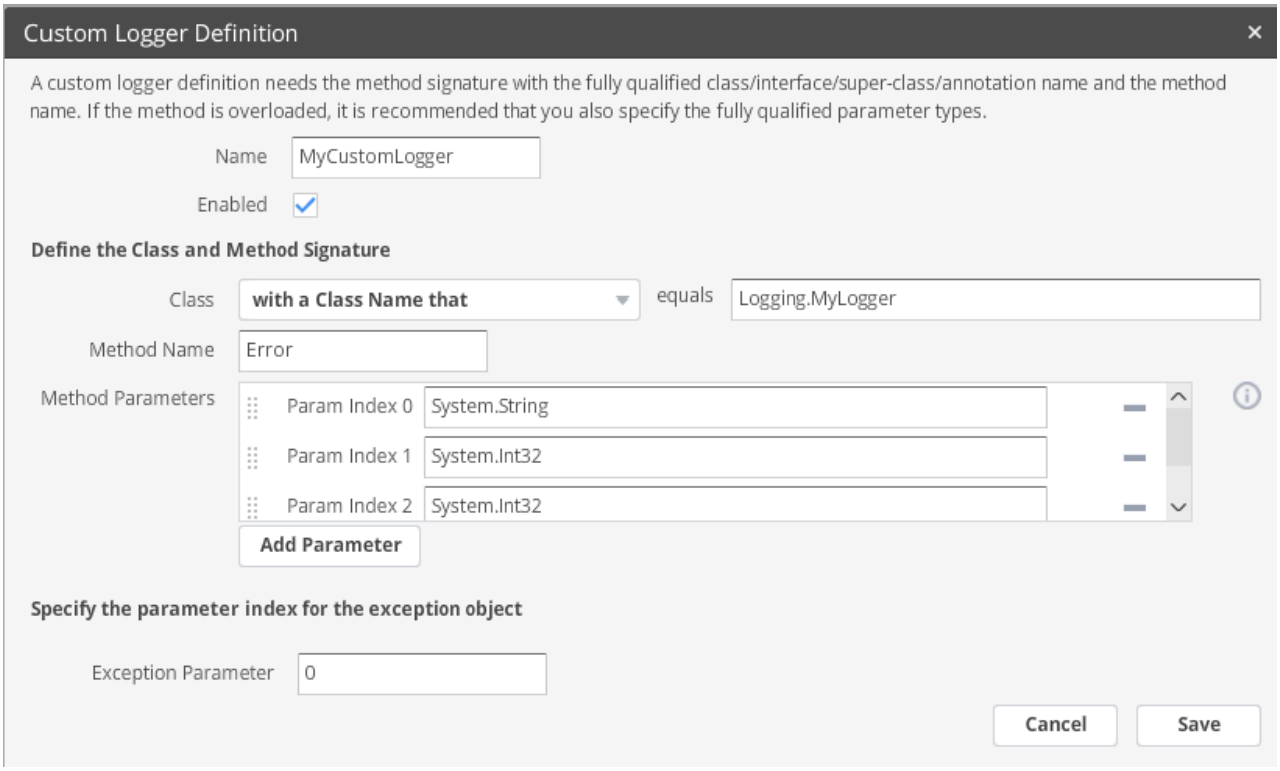
1. In the Controller UI, on the Error Detection configuration tab, click **Add Custom Logger Definition** in the error configuration window.
2. Enter a descriptive name for the custom logger definition and use the settings to identify the classname and method for the custom logger. For more information on how to use the AppDynamics UI to identify classes and methods, see Data Collectors.

3. For the **Method Parameter** field, add a parameter definition for each parameter in the signature of the method.
   If the method is overloaded, create a logger definition for each form of the overloaded method where you want to detect errors. Your custom method must accept at least one parameter, which should be the parameter that conveys the logged error or exception information.
4. For the **Exception Parameter** field, identify the parameter in the method signature that contains the exception object by index number (0-based).
   This parameter, identified by index, can be any type of object, including Arrays. If the object is not null, AppDynamics converts the object to a string, the result of which constitutes the error details. A business transaction is considered to be an error only if a non-null exception object is passed to the logger method.

Consider the following custom logger class and methods in a .NET application:

- Namespace and class = Logging.MyLogger
- logger.Error(string message, int param1)
- logger.Error(string message, int param1, int param2)

Notice that the Error method is overloaded. To capture errors logged in the second form of the method using the first parameter as the error to log. The following screenshot illustrates this configuration:



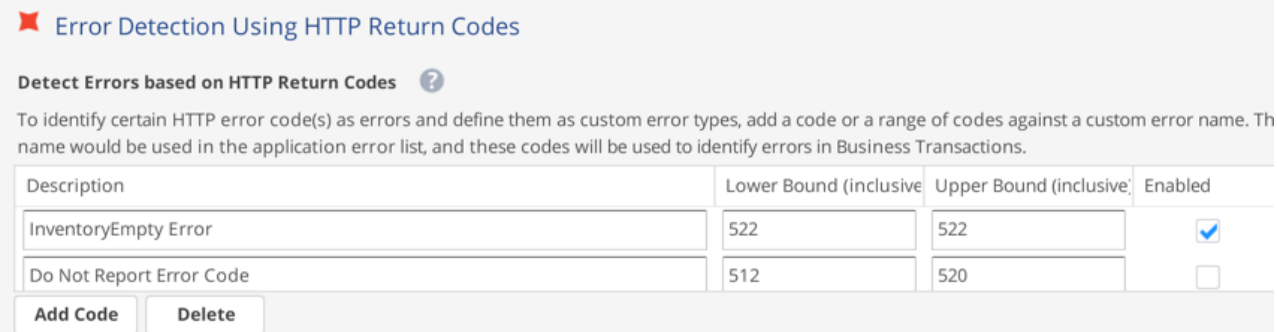
Notice that the exception parameter is set to 0, identifying the `string message` method parameter as the error message. When enabled, AppDynamics detects and reports the custom logger.

Detect HTTP Responses or Redirect Pages as Errors

In Java, .NET, Node.js, and Python you can configure errors based on HTTP response codes. If the error code is set as part of business transaction processing, the transaction is marked as an error.

By default, AppDynamics captures HTTP error codes from 400 to 505. HTTP response codes may convey errors that occur at the business level of an application. For example, in an ecommerce application, a 522 error might indicate that an item is out of stock. For this case, you may want to include the error as a default transaction error indicator.

To exclude a return code add them to the error detection list by creating a custom error code range and then disabling that error code by clearing its **Enabled** checkbox. This in effect excludes the error code as an error indicator.



For Java and .NET agents, you can specify custom redirect target error pages as error indicators. To specify a redirect page, click **Add Error Redirect Page** and add a name for the configuration and a regular expression to match the URL of the page, such as "AcmeErrorPage.jsp".

Ignore Exceptions and Log Messages as Error Indicators

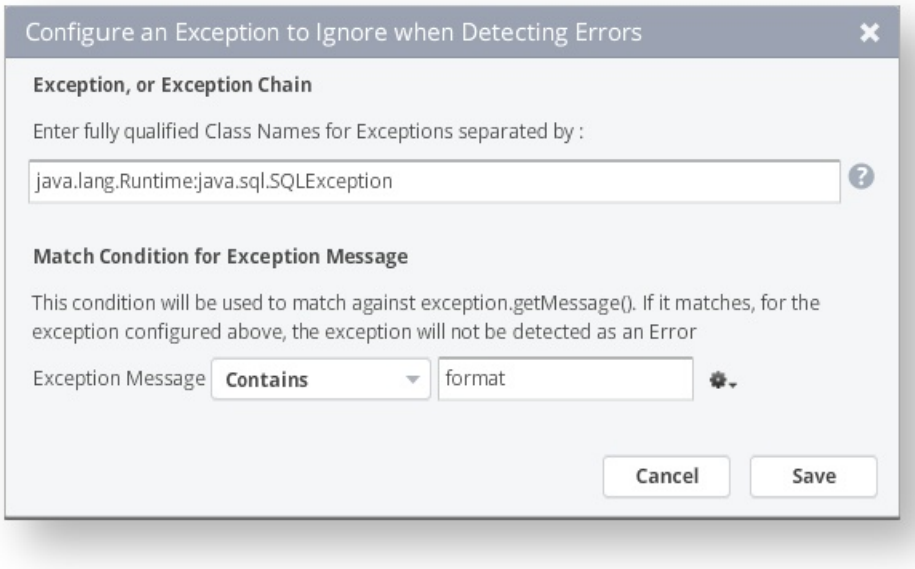
Certain types of exceptions, loggers or log messages as transaction error indicators may not reflect events that should be counted as transaction errors in your environment. They may include exceptions raised by application framework code or by user login failures.

When you configure an exception to be ignored, the agent detects the exception, increments the exception count, and displays the exception in Exceptions lists in the UI, but the business transaction in which the error is thrown is not considered an "error transaction" for monitoring purposes. The transaction snapshot would not show the exception in the Summary or Error Details section and the user experience for the transaction instance would be unaffected by the exception.

To configure exceptions for the agent to ignore, click the **Add New Exception to Ignore** button in the error configuration window.

Enter the class name of the exception to be ignored and the match condition for the exception message. For the match condition, if you do not need to filter for specific messages in the exception, select "Is Not Empty". If you want to filter for Null, select "Is Not Empty" and use the gear icon to select NOT, which, in effect, tests for "is empty".

The following example directs the agent to ignore java.lang.RuntimeExceptions that wrap a javax.sql.SQLException only when the exception.getMessage() call in the root exception contains the string "format". (Other types of java.lang.Runtime exceptions will not be ignored.)



When you define the class of an exception to ignore by an exception chain, the exception message against which the match condition is applied must be in the root exception of the chain. The match is not applied to any nested exceptions.

In .NET and Java, you can specify that errors logged with certain loggers or logger categories be ignored. Click **Add New Category/Logger to Ignore**.

Error Detection Notes by Platform

The following sections list error detection configuration and error monitoring considerations, if any, by application platform.

Error Configuration for .NET

By default, the agent can instrument calls using NLog and Log4Net. It can also gather information from System Trace and the Event Log. To instrument calls using other loggers, add a custom logger definition. See Configuring a Custom Logger.

Messages logged as higher than ERROR include severe levels such as CRITICAL or FATAL.

If you do *not* want system trace or event log errors to be monitored, disable them by checking the appropriate box. System trace errors are anything written as an error message to the Listeners collection via TraceError.

Event log errors are anything written to the EventLog when the type is set to error. For example:

```
myEventLog.WriteEntry(myMessage, EventLogEntryType.Error, myID);
```

Error Configuration for PHP

The PHP Agent instruments the PHP reporting facility. PHP applications can use trigger_error to report errors through that facility. PHP extensions and PHP itself can also use the PHP facility for reporting errors.

By default, the agent can check for PHP errors using multiple thresholds:

- If you select Error, the agent reports only messages and exceptions marked Error.
- If you select Warning, the agent reports only messages and exceptions marked Error and Warning.
- If you select Notice, the agent reports messages and exceptions marked Error, Warning and Notice.

Error Configuration for Node.js

The Node.js Agent reports exceptions thrown by the Node.js application or by Node.js itself.

You can configure certain exceptions or logged messages not to cause transactions to be reported as errors using the ‘Ignored Exceptions’ and ‘Ignored Messages’ lists. See Ignoring Exceptions and Log Messages as Error Indicators.

Error Configuration for Python Notes

By default, the Python Agent reports error transactions for unhanded exceptions, HTTP status codes greater than or equal to 400, and messages logged at ERROR or higher by the Python logging module.

If you do not want any logged errors to cause transactions to be reported as errors, clear the Mark Business Transaction as error check box. If you do not want to capture logged messages at all, clear the Detect Errors check box box.

You can configure certain exceptions or logged messages not to cause transactions to be reported as errors using the ‘Ignored Exceptions’ and ‘Ignored Messages’ lists. See Ignoring Exceptions and Log Messages as Error Indicators.