

# Zend\_CircuitBreaker - Artur Ejsmont

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

## Zend Framework: Zend\_CircuitBreaker Component Proposal

<b>Proposed Component Name</b>	Zend_CircuitBreaker
<b>Developer Notes</b>	<a href="http://framework.zend.com/wiki/display/ZFDEV/Zend_CircuitBreaker">http://framework.zend.com/wiki/display/ZFDEV/Zend_CircuitBreaker</a>
<b>Proposers</b>	Artur Ejsmont
<b>Zend Liaison</b>	TBD
<b>Revision</b>	1.0 - 20 November 2010: Initial Draft. (wiki revision: 21)

## Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

## 1. Overview

Circuit Breaker is a component that helps discovering external resources failures and adapting application behavior.

If my database is overloaded and times out after 30s or a web service is broken i do not want to waste time connecting to it. I prefer to use long lived cache or even show a user friendly message instead.

Circuit breaker helps by encapsulating this logic and allowing developers to simply ask "is my resource available?" before using it. CB manages statistical data to give the appropriate answer.

Statistical data is stored using persistence adapter as an internal data structure. This information is gathered after every request to the external service so that CB would know is the external service healthy or not. Storage and retrieval of the data has to be very fast (apc/memcached preferred).

## 2. References

- [Wikipedia Entry](#)
- [More details on my site](#)

### 3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** allow configuration of thresholds per service name.
- This component **will** allow configuration via array/config object.
- This component **will** allow replacement of persistence storage adapters.
- This component **will** provide persistence storage adapters for zend cache, dummy and database.
- This component **will** keep very simple API.
- This component **will** be ported to ZF2 as soon as zend cache 2 is fully implemented.
  
- This component **will not** be an singleton
- This component **will not** depend on other components unless specified explicitly by user - to allow standalone usage.

### 4. Dependencies on Other Framework Components

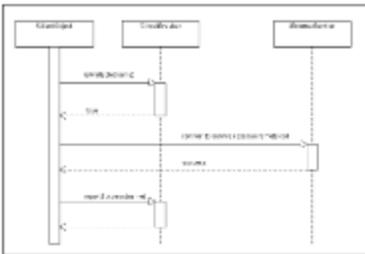
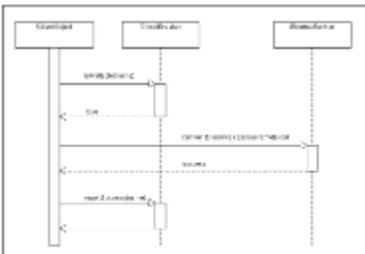
- Zend\_Exception
- Zend\_Cache - optional, you can provide your own Zend\_CircuitBreaker\_Storage\_Interface implementations.

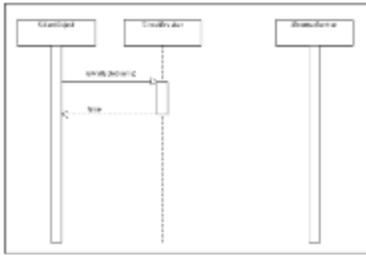
### 5. Theory of Operation

This component should be used close to the service call implementation. To decrease code duplication and end user involvement it would be beneficial to create decorators (or some extensions) to database and web service components. This way calls to Circuit breaker could be handed by Zend Framework. It is not required for a standalone use though.

Sequence:

- User calls Circuit breaker to ask is service named 'ABC' available?
- Circuit breaker responds true if service is available or false if its considered unavailable
- User calls service based on the advice
- After service call user reports back to circuit breaker to let it know was service call successful or not.





This way CB can set thresholds of amount of failed requests in period of time. If requests were failing lately CB can start advising that service is unavailable. Therefore user code can skip service call and branch execution straight to error handling. User can also decide to use long lived cache in this scenario or take other actions necessary.

Diagram of class dependencies can be seen below. Circuit breaker delegates persistence to selected adapter type. It can also use array decorator to group all service statistics into one array - to reduce persistence storage load/save calls.



Service back online discovery.

Circuit breaker comes with an important feature of probing for service availability. Once service is recognized as unavailable CB will allow single thread to try to use it based on configuration. This way you can configure retry to happen after 1,5,60 minutes or whatever value in seconds. CB will make sure just one thread will try to connect, if it succeeds it will allow rest of the threads to connect too. Otherwise it will keep service locked till next retry timeout.

## 6. Milestones / Tasks

Component is in usable state available (for now) [circuit breaker code on my website](#). There are full implementations of core classes and some test coverage.

- Milestone 1: Decide should exceptions be disabled (should CB throw store exception) or maybe add store exception capturing decorator and config option for it?
- Milestone 2: Decide if APC handler could be included in component to have no dependency on zend cache - even that some (minimal) duplication occurs.
- Milestone 3: Initial documentation exists but more is needed
- Milestone 4: Check component somewhere into git (i don't know where should it be available yet)

## 7. Class Index

- Zend\_CircuitBreaker\_Interface
- Zend\_CircuitBreaker
- Zend\_CircuitBreaker\_Storage\_Interface
- Zend\_CircuitBreaker\_Storage\_Exception
- Zend\_CircuitBreaker\_Storage\_Adapter\_Dummy
- Zend\_CircuitBreaker\_Storage\_Adapter\_ZendCacheBackend
- Zend\_CircuitBreaker\_Storage\_Decorator\_Array
- Zend\_CircuitBreaker\_Storage\_Adapter\_Apc (optionally for performance and simplicity)

## 8. Use Cases

## 9. Class Skeletons

]]</ac:plain-text-body></ac:macro>

]]</ac:plain-text-body></ac:macro>

<p>Unable to render embedded object: File (classes) not found.</p>