Exakat Documentation

Release 1

Damien Seguy

Oct 01, 2020

Contents

Contents:

CHAPTER 1

Introduction

This is the documentation of the Exakat engine, version 2.1.9 (Build 1153), on Tue, 22 Sep 2020 06:05:09 +0000.

1.1 What is Exakat ?

Exakat is a tool for analyzing, reporting and assessing PHP code source efficiently and systematically. Exakat processes PHP 5.2 to 7.4 and 8.0 code, as well as reporting on security, performance, code quality, migration.

Exakat reads the code, builds an AST and several dependency graphs, then indexes all of it in a graph database. From there, exakat runs analysis, collecting potential errors and descriptive information about the code. Finally, exakat produces reports, both for humans and machines.

1.2 Exakat Use Cases

1.2.1 Code quality

Exakat detects hundreds of issues in PHP code : dead code, incompatible calls, undefined calls, illogical expressions, etc. Exakat is built for PHP, and cover common mistakes.

1.2.2 PHP version migration

Every PHP middle version is a migration by itself : based on the manual and common practices, exakat find both backward incompatibilities, that prevent migration, and new features, that makes code modern.

Exakat review code for minor version, and spot bug fixes that may impact the code.

1.2.3 Framework code quality

Common best practices and recommendations for specific plat-forms like Wordpress, CakePHP or Zend Framework are covered.

1.2.4 PHP configurations

Exakat detects several specialized analyzes, for Web security : making the code more secure online; PHP performances : allowing faster execution.

1.2.5 Security, performances, testability

Exakat has several specialized analyzes, for Web security : making the code more secure online; PHP performances : allowing faster execution; Testability : targeting the common pitfalls that makes code less testable.

1.2.6 Feature inventories

When auditing code, it is important to have a global view. Exakat collects all PHP features (magic functions, any operator, special functions or patterns) and represents them in one report, giving auditors a full view.

Exakat inventories all literals for later review, helping with the magic number syndrome and any data refactoring.

1.3 Exakat compared to others

1.3.1 Code sniffer

Automated coding standard violation detection for PHP review the code for syntax layout. Exakat is not a coding standard detection tool, as it focuses on bug finding, rather than coding layout.

While checking for coding standard, some bugs may be detected, and when checking for bugs, some coding standards may be found too.

Using AST, dependency graphs and knowledge databases, Exakat reviews the code, checks its potential usage and mis-usage. Exakat doesn’t take any presentation nor comments into accounts : only functions, variables and their effects.

1.3.2 Phan, PHPstan, PHP

PHP code quality checks, based on type compatibility, and structure definitions. Exakat shares AST style analysis but it goes a bit further by including common mistakes and actual PHP features detections.

1.3.3 PHP7mar, PHP7cc

Code review for PHP 5 to migrate to PHP 7. Exakat covers every middle version from PHP 5.3 to PHP 7.3.

1.3.4 PHP-ci, Jenkins, Grumphp

Continuous integration and code quality management check the code by running code quality tools and collecting all the reported informations. Exakat is a good companion for those tools.

Exakat provides machine readable format reports, such as json, xml, text that may be consumed by CI. Exakat provides also human readable format, such as HTML, for interactive review of the reports, and a longer usage life span.

1.4 Exakat ecosystem

Exakat is an Open Source tool. The code is available on Github.com/exakat/exakat, as Docker image and Vagrant file. It is also available as a phar download.

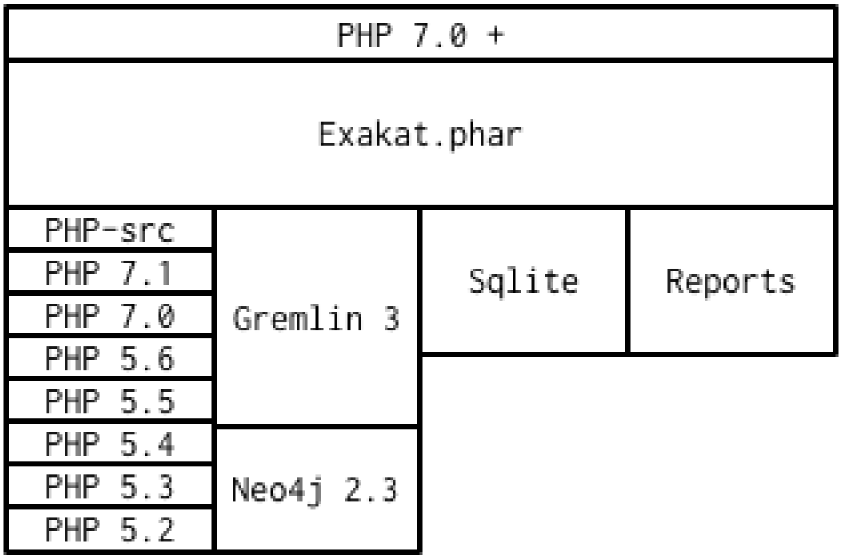
Exakat cloud is a SaaS platform, offering exakat audits on code, anytime, at reduced cost.

Exakat SAS is a Service company, providing consulting and training services around automated analysis and code quality for PHP.

1.5 Architecture

Exakat relies on PHP to lint and tokenize the target code; a graph database to process the AST and the tokens; a SQLITE 3 database to store the results and produce the various reports.

Exakat itself runs on PHP 7.2, with a short selection of extensions. It is tested with PHP 7.0 and 7.3.



Source code is imported into exakat using VCS client, like git, SVN, mercurial, tar, zip, bz2 or even symlink. Only reading access is actually required : the code is never modified in any way.

At least one version of PHP have to be used, and it may be the same running Exakat. Only one version is used for analysis and it may be different from the running PHP version. For example, exakat may run with PHP 7.2 but audit code with PHP 5.6. Extra versions of PHP are used to provide compilations reports. PHP middle versions may be configured separately. Minor versions are not important, except for edge cases.

The gremlin server is used to query the source code. Once analyzes are all finished, the results are dumped into a SQLITE database and the graph may be removed. Reports are build from the SQLITE database.

CHAPTER 2

Exakat features

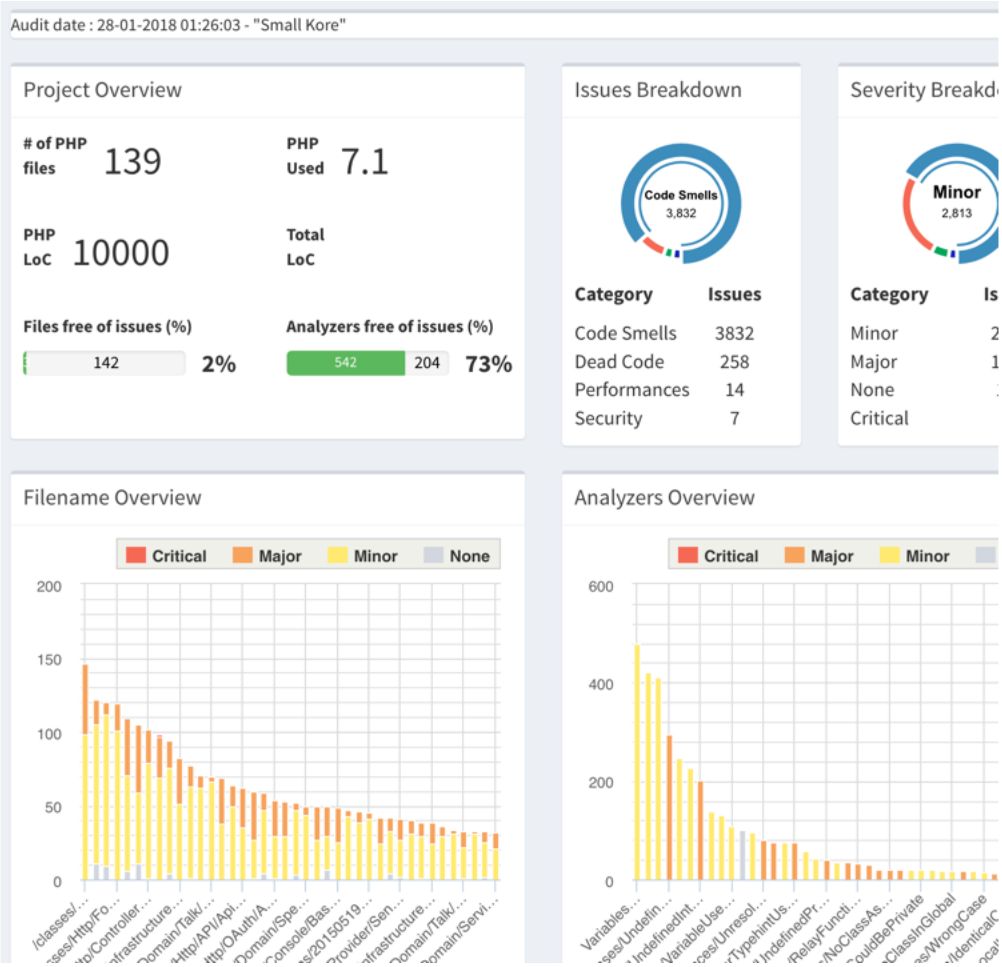
2.1 Features list

- 412 analyzers
- Compatible with PHP 5.2 to 8.0-dev
- Migration guide from 5.2 to 7.4 and 8.0-dev
- Modernize your code
- List bug fixes for your code
- appinfo(): the list of PHP features
- List PHP directives that impact your code
- Framework and application support
- Hierarchy Diagrams
- Code visualizations

2.2 412 analyzers

There are currently 412 different analyzers that check the PHP code to report code smells. Analyzers are inspired by PHP manual, migration documents, community good practices, computer science or simple logic.

Some of them track rare occurrences, and some are frequent. Some track careless mistakes and some are highly complex situations. In any case, exakat has your back, and will warn you.



Audit date : 28-01-2018 01:26:03 - "Small Kore"

Project Overview

# of PHP files	139	PHP Used	7.1
PHP LoC	10000	Total LoC	

Files free of issues (%): 142 — 2%

Analyzers free of issues (%): 542 — 204 — 73%

Issues Breakdown

Category	Issues
Code Smells	3832
Dead Code	258
Performances	14
Security	7

Severity Breakd

Category	Is
Minor	2
Major	1
None	
Critical	

2.3 Compatible with PHP 5.2 to 8.0-dev

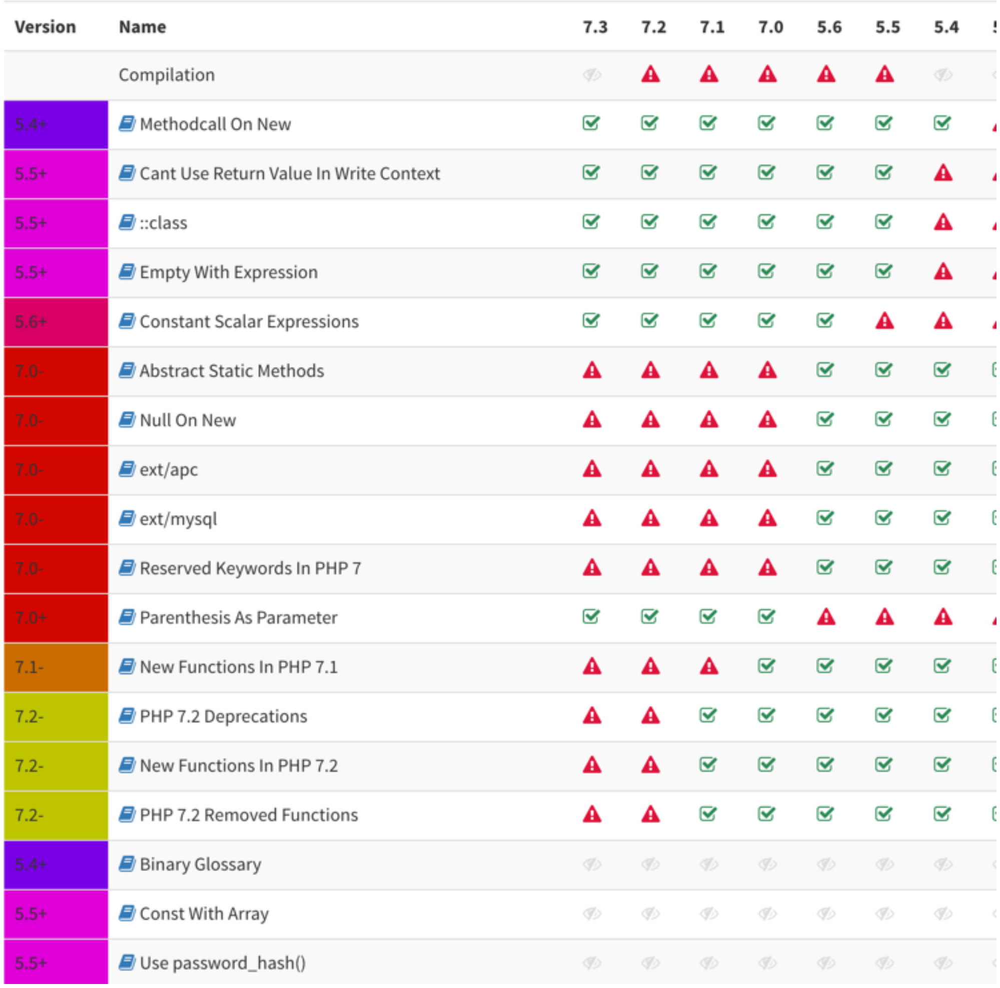
The Exakat engine audits code with PHP versions that range from PHP 5.2 to PHP 8.0-dev.

The Exakat engine itself runs on PHP 7.x+ and is regularly checked on those versions. It is possible to run Exakat on 7.2 and audit a code with PHP 5.6.

2.4 Migration guide from 5.2 to 8.0-dev

Every middle version of PHP comes with its migration guide from the manual, and from community's feedback. Incompatibilities are included as analyzers in Exakat, and report everything they can find that may prevent you from moving to the newer version.

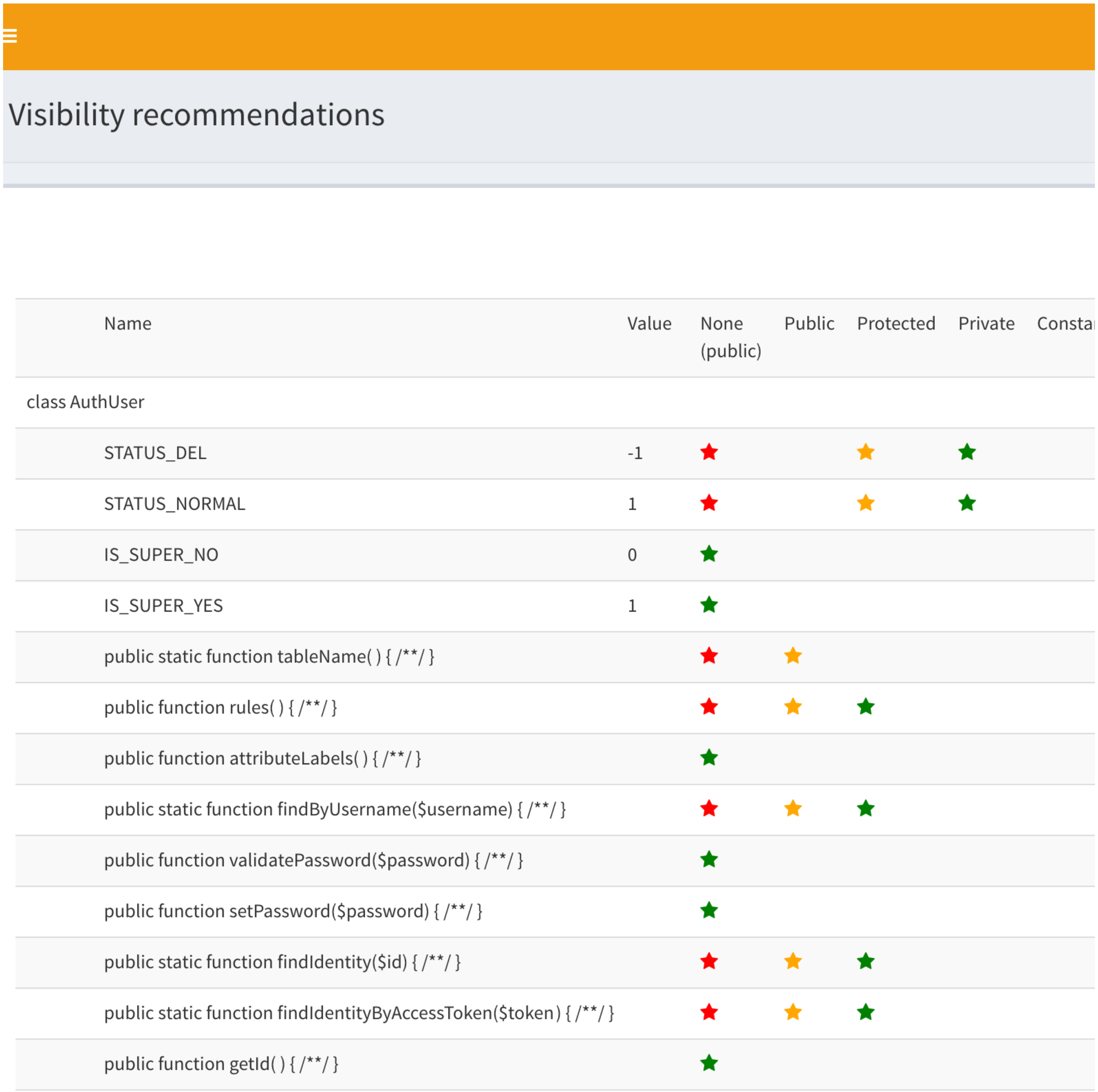
Although they won't catch it all, they do reduce the amount of unexpected surprises by a lot.



Version	Name	7.3	7.2	7.1	7.0	5.6	5.5	5.4
	Compilation	⊘	⚠	⚠	⚠	⚠	⚠	⊘
5.4+	Methodcall On New	✓	✓	✓	✓	✓	✓	✓
5.5+	Cant Use Return Value In Write Context	✓	✓	✓	✓	✓	✓	⚠
5.5+	::class	✓	✓	✓	✓	✓	✓	⚠
5.5+	Empty With Expression	✓	✓	✓	✓	✓	✓	⚠
5.6+	Constant Scalar Expressions	✓	✓	✓	✓	✓	⚠	⚠
7.0-	Abstract Static Methods	⚠	⚠	⚠	⚠	✓	✓	✓
7.0-	Null On New	⚠	⚠	⚠	⚠	✓	✓	✓
7.0-	ext/apc	⚠	⚠	⚠	⚠	✓	✓	✓
7.0-	ext/mysql	⚠	⚠	⚠	⚠	✓	✓	✓
7.0-	Reserved Keywords In PHP 7	⚠	⚠	⚠	⚠	✓	✓	✓
7.0+	Parenthesis As Parameter	✓	✓	✓	✓	⚠	⚠	⚠
7.1-	New Functions In PHP 7.1	⚠	⚠	⚠	✓	✓	✓	✓
7.2-	PHP 7.2 Deprecations	⚠	⚠	✓	✓	✓	✓	✓
7.2-	New Functions In PHP 7.2	⚠	⚠	✓	✓	✓	✓	✓
7.2-	PHP 7.2 Removed Functions	⚠	⚠	✓	✓	✓	✓	✓
5.4+	Binary Glossary	⊘	⊘	⊘	⊘	⊘	⊘	⊘
5.5+	Const With Array	⊘	⊘	⊘	⊘	⊘	⊘	⊘
5.5+	Use password_hash()	⊘	⊘	⊘	⊘	⊘	⊘	⊘

2.5 Modernize your code

Migrations are too often considered over when incompatibilities are removed. In fact, the best is still to come : using the new features. Or, using the new features from previous versions, that were forgotten. Exakat dedicates a whole category of suggestions to modern PHP features that should be used now.



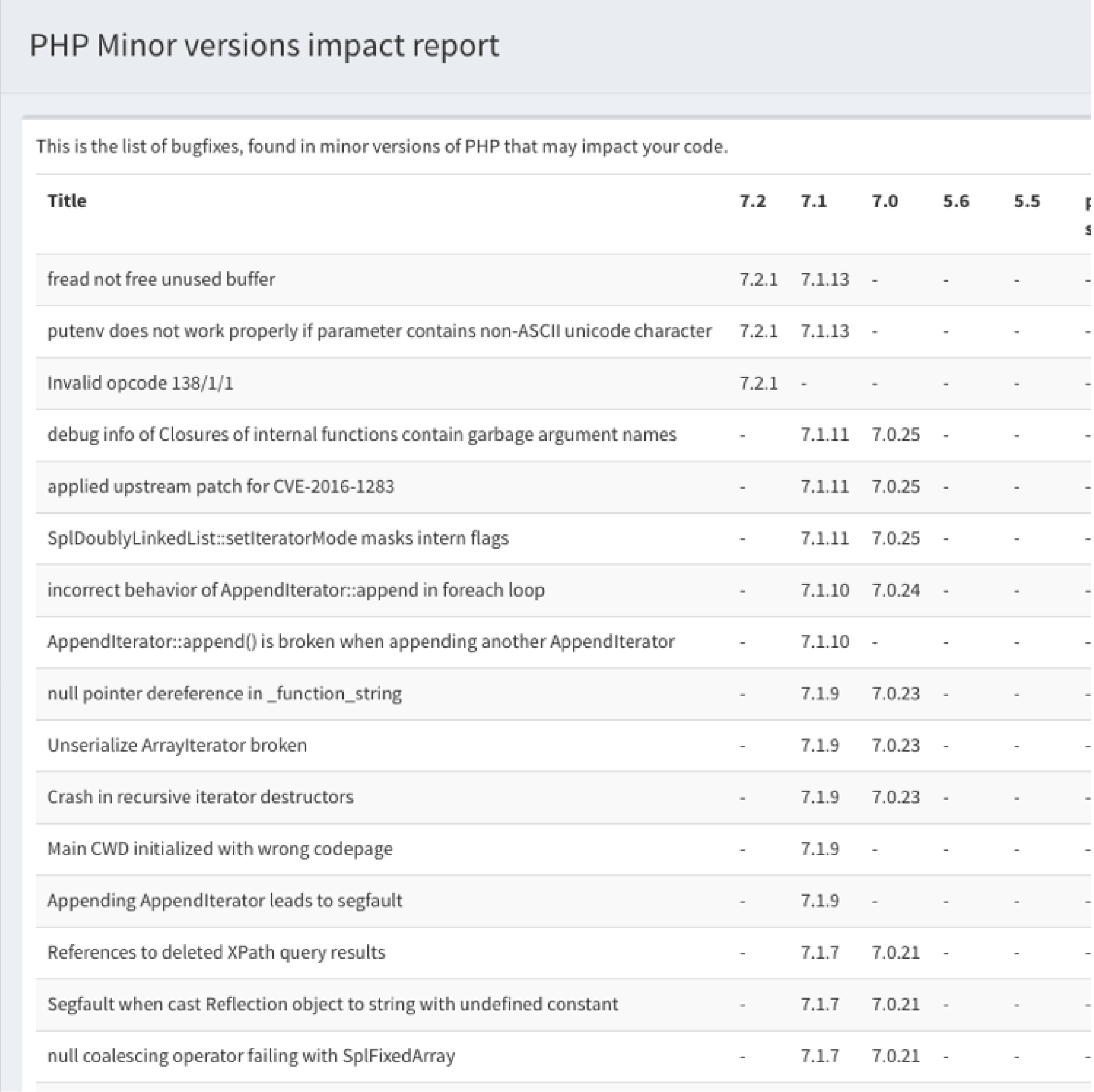
Visibility recommendations

Name	Value	None (public)	Public	Protected	Private	Consta
class AuthUser						
STATUS_DEL	-1	★		★	★	
STATUS_NORMAL	1	★		★	★	
IS_SUPER_NO	0	★				
IS_SUPER_YES	1	★				
public static function tableName( ) { /**/ }		★	★			
public function rules( ) { /**/ }		★	★	★		
public function attributeLabels( ) { /**/ }		★				
public static function findByUsername($username) { /**/ }		★	★	★		
public function validatePassword($password) { /**/ }		★				
public function setPassword($password) { /**/ }		★				
public static function findIdentity($id) { /**/ }		★	★	★		
public static function findIdentityByAccessToken($token) { /**/ }		★	★	★		
public function getId( ) { /**/ }		★				

2.6 Bug fixes that impact the code

Every minor version of PHP comes with bug fixes and modifications at the function level. Some special situations are better handled, and that may have impact in your code. Every modified function, class, trait or interface that is also found in your code is reported here, giving a good overview of the impact of every minor version.

Safe bet : keep up to date!



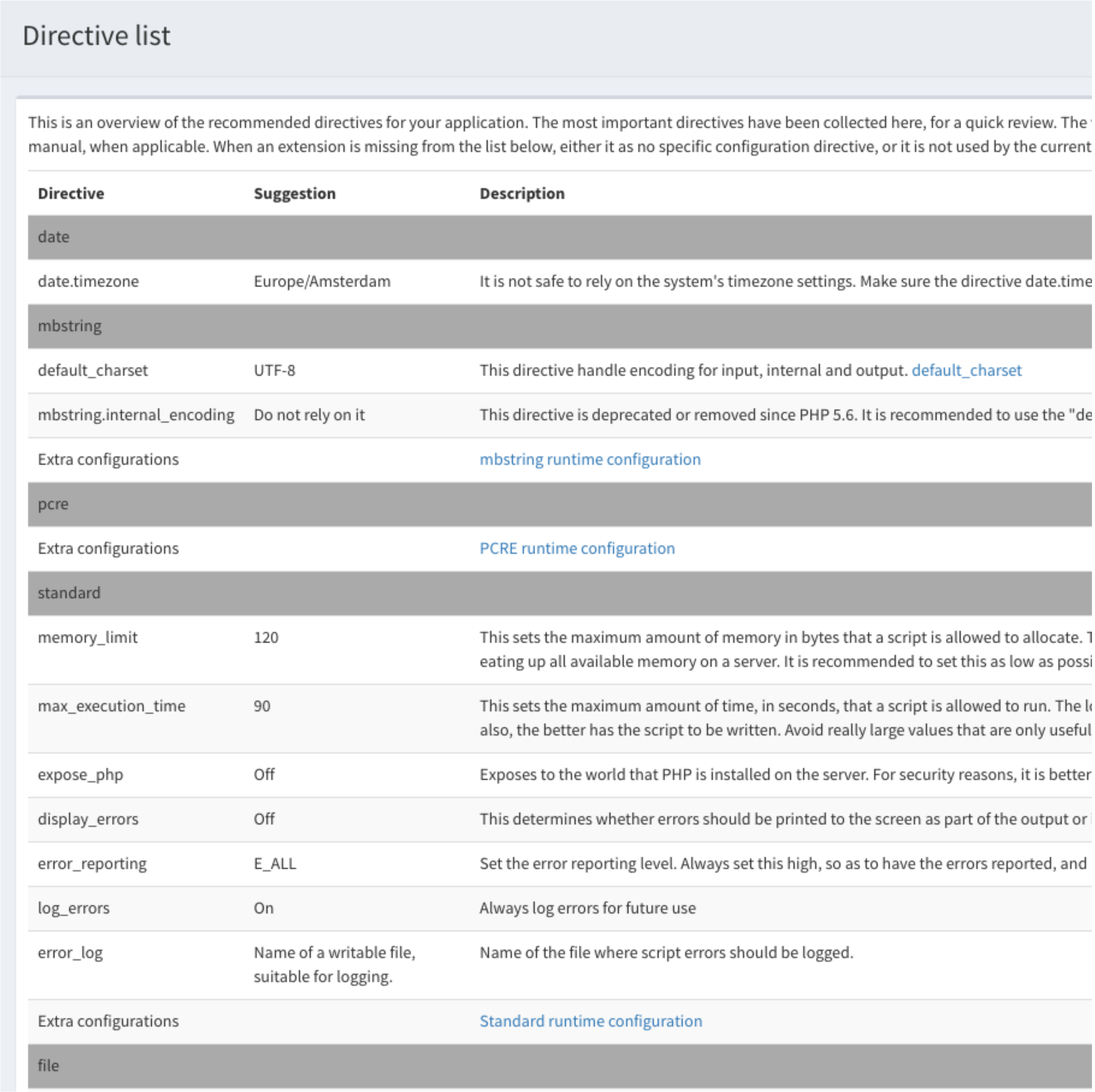
PHP Minor versions impact report

This is the list of bugfixes, found in minor versions of PHP that may impact your code.

Title	7.2	7.1	7.0	5.6	5.5	p s
fread not free unused buffer	7.2.1	7.1.13	-	-	-	-
putenv does not work properly if parameter contains non-ASCII unicode character	7.2.1	7.1.13	-	-	-	-
Invalid opcode 138/1/1	7.2.1	-	-	-	-	-
debug info of Closures of internal functions contain garbage argument names	-	7.1.11	7.0.25	-	-	-
applied upstream patch for CVE-2016-1283	-	7.1.11	7.0.25	-	-	-
SplDoublyLinkedList::setIteratorMode masks intern flags	-	7.1.11	7.0.25	-	-	-
incorrect behavior of AppendIterator::append in foreach loop	-	7.1.10	7.0.24	-	-	-
AppendIterator::append() is broken when appending another AppendIterator	-	7.1.10	-	-	-	-
null pointer dereference in _function_string	-	7.1.9	7.0.23	-	-	-
Unserialize ArrayIterator broken	-	7.1.9	7.0.23	-	-	-
Crash in recursive iterator destructors	-	7.1.9	7.0.23	-	-	-
Main CWD initialized with wrong codepage	-	7.1.9	-	-	-	-
Appending AppendIterator leads to segfault	-	7.1.9	-	-	-	-
References to deleted XPath query results	-	7.1.7	7.0.21	-	-	-
Segfault when cast Reflection object to string with undefined constant	-	7.1.7	7.0.21	-	-	-
null coalescing operator failing with SplFixedArray	-	7.1.7	7.0.21	-	-	-

2.7 appinfo(): the list of PHP features

Do you know the PHP features that your application rely upon ? Recursivité, reflexion, backticks or anonymous classes ? Exakat collect all those features, and sum them up in one nice table, so you know all of it.



Directive list

This is an overview of the recommended directives for your application. The most important directives have been collected here, for a quick review. The
manual, when applicable. When an extension is missing from the list below, either it as no specific configuration directive, or it is not used by the current

Directive	Suggestion	Description
date		
date.timezone	Europe/Amsterdam	It is not safe to rely on the system's timezone settings. Make sure the directive date.time
mbstring		
default_charset	UTF-8	This directive handle encoding for input, internal and output. default_charset
mbstring.internal_encoding	Do not rely on it	This directive is deprecated or removed since PHP 5.6. It is recommended to use the "de
Extra configurations		mbstring runtime configuration
pcre		
Extra configurations		PCRE runtime configuration
standard		
memory_limit	120	This sets the maximum amount of memory in bytes that a script is allowed to allocate. T eating up all available memory on a server. It is recommended to set this as low as possi
max_execution_time	90	This sets the maximum amount of time, in seconds, that a script is allowed to run. The l also, the better has the script to be written. Avoid really large values that are only useful
expose_php	Off	Exposes to the world that PHP is installed on the server. For security reasons, it is better
display_errors	Off	This determines whether errors should be printed to the screen as part of the output or
error_reporting	E_ALL	Set the error reporting level. Always set this high, so as to have the errors reported, and
log_errors	On	Always log errors for future use
error_log	Name of a writable file, suitable for logging.	Name of the file where script errors should be logged.
Extra configurations		Standard runtime configuration
file		

2.8 List of significant PHP directives

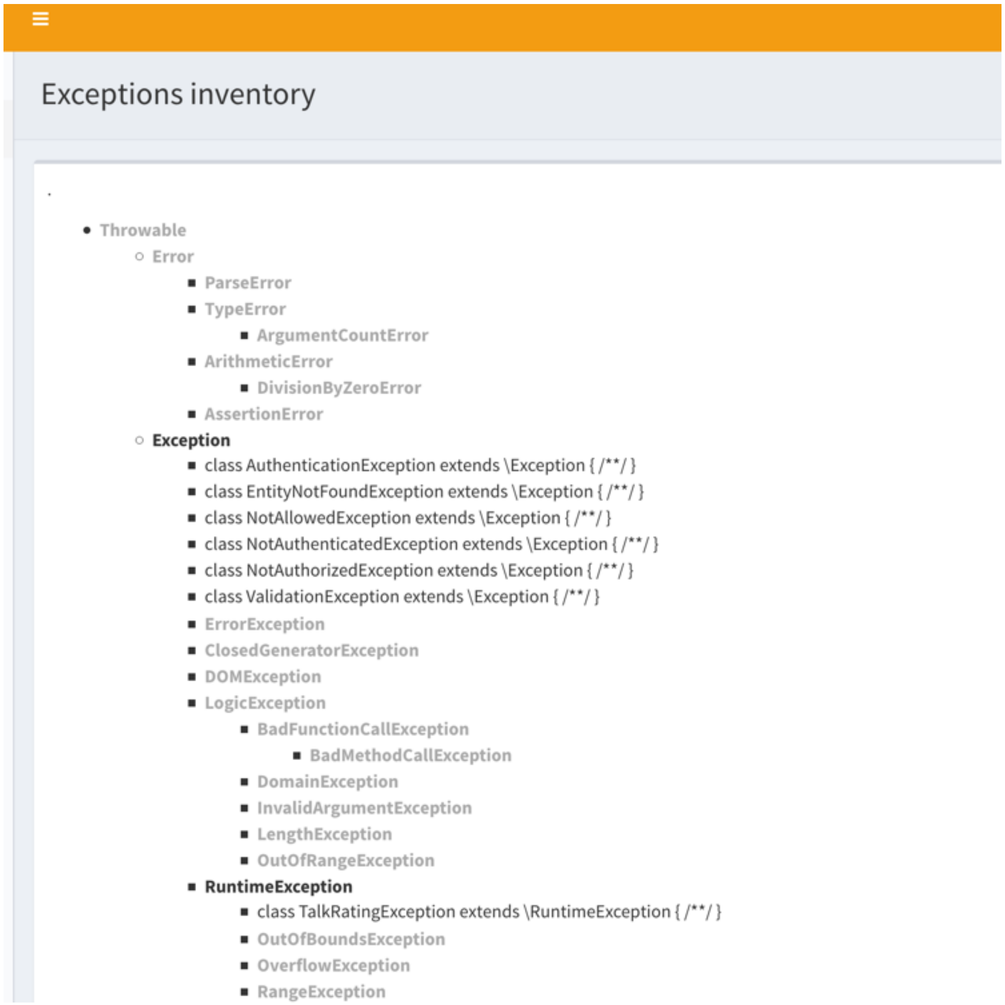
Exakat recommends which PHP directives to check while preparing your code for production. If ‘memory_limit’ is an ever green, may be ‘post_max_size’ (linked to file_upload), or assertions shouldn’t be forgotten. Based on feature and extension usage, it also list the most important directives, and leads you to the full manual list, in case you want to fine tune it to the max. Use it as a reminder.

2.9 Framework and application support

Exakat provides support for framework and application specific rules. Supported frameworks includes Cakephp, Codeigniter, Drupal, Laravel, Melis, Slim, Symfony, Wordpress and Zend Framework

2.10 Hierarchy Diagrams

Exakat documents the code automatically with several diagrams, such as : * UML class diagramm, based on inheritance (classes), usage (traits) and implementations (interfaces), grouped by namespaces. * The Exceptions tree * The traits tree and the trait matrix

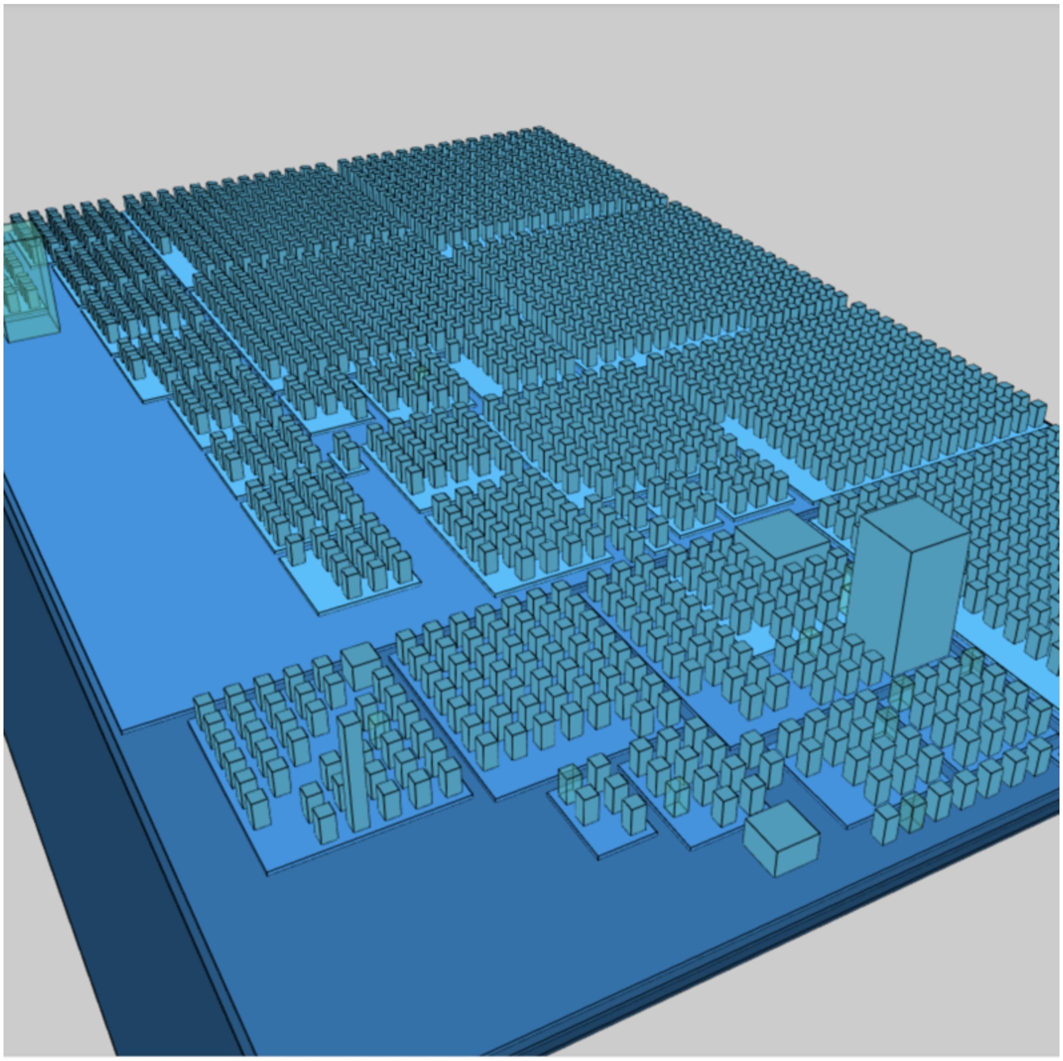


Exceptions inventory

- Throwable
    - Error
        - ParseError
        - TypeError
            - ArgumentCountError
        - ArithmeticError
            - DivisionByZeroError
        - AssertionError
    - **Exception**
        - class AuthenticationException extends \Exception { /**/ }
        - class EntityNotFoundException extends \Exception { /**/ }
        - class NotAllowedException extends \Exception { /**/ }
        - class NotAuthenticatedException extends \Exception { /**/ }
        - class NotAuthorizedException extends \Exception { /**/ }
        - class ValidationException extends \Exception { /**/ }
        - ErrorException
        - ClosedGeneratorException
        - DOMException
        - LogicException
            - BadFunctionCallException
                - BadMethodCallException
            - DomainException
            - InvalidArgumentException
            - LengthException
            - OutOfRangeException
        - **RuntimeException**
            - class TalkRatingException extends \RuntimeException { /**/ }
            - OutOfBoundsException
            - OverflowException
            - RangeException

2.11 Code visualizations

Exakat documents the code automatically with several diagrams, such as : a full UML class diagramm, based on inheritance (classes), usage (traits) and implementations (interfaces), grouped by namespaces.



CHAPTER 3

Installation

3.1 Summary

- *Requirements*
- *Quick installation with exakat.phar*
- *Quick installation with OSX*
- *Full installation with Debian/Ubuntu*
- *Quick installation with Debian/Ubuntu*
- *Installation guide with Composer*
- *Installation guide with Docker*
- *Installation guide as Github Action*
- *Installation guide for optional tools*

3.2 Requirements

Exakat relies on several parts. Some are necessary and some are optional.

Basic requirements :

- exakat.phar, the main code.
- Gremlin server : exakat uses this graph database and the Gremlin 3 traversal language. Currently, only Gremlin Server is supported, with the tinkergraph and neo4j storage engine. Version 3.4.x is the recommended version, while version 3.3.x are still supported. Gremlin version 3.2.* are unsupported.
- Java 8.x. Java 9.x/10.x will be supported later. Java 7.x was used, but is not actively supported.
- PHP 7.4 to run. PHP 7.4 is recommended, PHP 7.2 or later are possible. This version requires the PHP extensions curl, hash, phar, sqlite3, tokenizer, mbstring and json.

Optional requirements :

- PHP 5.2 to 8.0-dev for analysis purposes. Those versions only require the ext/tokenizer extension.
- VCS (Version Control Software), such as Git, SVN, bazaar, Mercurial. They all are optional, though git is recommended.
- Archives, such as zip, tgz, tbz2 may also be opened with optional helpers (See *Installation guide for optional tools*).

OS requirements : Exakat has beed tested on OSX, Debian and Ubuntu (up to 20.04). Exakat should work on Linux distributions, may be with little work. Exakat hasn't been tested on Windows at all.

For installation, curl or wget, and zip are needed.

3.3 Download Exakat

You can download exakat directly from https://www.exakat.io/.

This server also provides older versions of Exakat. It is recommended to always download the last version, which is available with https://www.exakat.io/versionss/index.php?file=latest.

For each version, MD5 and SHA256 signatures are available. The downloaded MD5 must match the one in the related .md5 file. The .md5 also has the version number, for extra check.

```
curl -o exakat.phar 'https://www.exakat.io/versions/index.php?file=latest'

curl -o exakat.phar.md5 'https://www.exakat.io/versions/index.php?file=latest.md5'
//19485adb7d43b43f7c01b7153ae82881  exakat-2.0.0.phar
md5sum exakat.phar.md5
// Example :
//19485adb7d43b43f7c01b7153ae82881  exakat.phar

curl -o exakat.phar.sha256 'https://www.exakat.io/versions/index.php?file=latest.
↪sha256'
//d838c9ec9291e15873137693da2a0038a67c2f15c2282b89f09f27f23d24d27f  exakat-2.0.0.phar
sha256sum exakat.phar.md5
// Example :
//d838c9ec9291e15873137693da2a0038a67c2f15c2282b89f09f27f23d24d27f  exakat.phar

// Check with GPG signature
curl -o exakat.sig 'https://www.exakat.io/versions/index.php?file=latest.sig'
// Optional step : Download the Key
gpg --recv-keys 5EDF7EA4
// Check with GPG signature
gpg --verify exakat.sig exakat.phar
// Good result :
//gpg: Signature made Tue Nov  5 07:48:34 2019 CET using RSA key ID 5EDF7EA4
//gpg: Good signature from "Seguy Damien <damien.seguy@gmail.com>" [ultimate]
```

3.4 Quick installation with exakat.phar

3.4.1 OSX installation with tinkergraph 3.4.8

Exakat.phar includes its own installation script, as long as PHP is available. Exakat will then check different pre-requisites, and proceed to install some of the last elements.

Exakat checks for Java and Zip installations. Then, it downloads tinkergraph and the Neo4j plugin from exakat.io and runs the *doctor* command.

The script is based on the one displayed on the next section.

You can use the *install* command this way :

```
mkdir exakat
cd exakat
curl -o exakat.phar 'https://www.exakat.io/versions/index.php?file=latest'
php exakat.phar install
```

3.5 Quick installation with OSX

Paste the following commands in a terminal prompt. It downloads Exakat, and installs tinkerpop version 3.4.8. PHP 7.0 or more recent, curl, homebrew are required.

3.5.1 OSX installation with tinkergraph 3.4.8

This is the installation script for Exakat and tinkergraph 3.4.8.

```
mkdir exakat
cd exakat
curl -o exakat.phar 'https://www.exakat.io/versions/index.php?file=latest'
curl -o apache-tinkerpop-gremlin-server-3.4.8-bin.zip 'https://www.exakat.io/versions/
↪apache-tinkerpop-gremlin-server-3.4.8-bin.zip'
unzip apache-tinkerpop-gremlin-server-3.4.8-bin.zip
mv apache-tinkerpop-gremlin-server-3.4.8 tinkergraph
rm -rf apache-tinkerpop-gremlin-server-3.4.8-bin.zip

# Optional : install neo4j engine.
cd tinkergraph
./bin/gremlin-server.sh install org.apache.tinkerpop neo4j-gremlin 3.4.8
cd ..

php exakat.phar doctor
```

3.5.2 OSX installation troubleshooting

It has be reported that installation fails on OSX 10.11 and 10.12, with error similar to ‘Error grabbing Grapes’. To fix this, use the following in command line :

```
rm -r ~/.groovy/grapes/
rm -r ~/.m2/
```

They remove some files for grapes, that it will rebuild later. Then, try again the optional install instructions.

3.6 Full installation with Debian/Ubuntu

The following commands are an optional pre-requisite to the Quick installation guide, that just follows. If something is missing in the next section, check with this section that all has beed installed correctly.

```
//// Installing PHP from sury.org
apt update
apt install apt-transport-https lsb-release ca-certificates

wget -O /etc/apt/trusted.gpg.d/php.gpg https://packages.sury.org/php/apt.gpg
sh -c 'echo "deb https://packages.sury.org/php/ $(lsb_release -sc) main" > /etc/apt/
↪sources.list.d/php.list'
apt update

apt-get install php7.2 php7.2-common php7.2-cli php7.2-curl php7.2-json php7.2-
↪mbstring php7.2-sqlite3

//// Installing Java JDK
echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee /etc/
↪apt/sources.list.d/webupd8team-java.list
echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee -a /
↪etc/apt/sources.list.d/webupd8team-java.list
apt-get update

echo debconf shared/accepted-oracle-license-v1-1 select true | debconf-set-selections
echo debconf shared/accepted-oracle-license-v1-1 seen true | debconf-set-selections
DEBIAN_FRONTEND=noninteractive  apt-get install -y --force-yes oracle-java8-installer␣
↪oracle-java8-set-default

//// Installing other tools
apt-get update && apt-get install -y --no-install-recommends git subversion mercurial␣
↪lsof unzip
```

3.7 Quick installation with Debian/Ubuntu

3.7.1 Debian/Ubuntu installation with Tinkergraph 3.4.8

Paste the following commands in a terminal prompt. It installs Exakat most recent version with Tinkergraph 3.4.8. PHP 7.3 (7.0 or more recent), wget and unzip are expected.

```
mkdir exakat
cd exakat
wget -O exakat.phar https://www.exakat.io/versions/index.php?file=latest
wget -O apache-tinkerpop-gremlin-server-3.4.8-bin.zip 'https://www.exakat.io/versions/
↪apache-tinkerpop-gremlin-server-3.4.8-bin.zip'
unzip apache-tinkerpop-gremlin-server-3.4.8-bin.zip
mv apache-tinkerpop-gremlin-server-3.4.8 tinkergraph
rm -rf apache-tinkerpop-gremlin-server-3.4.8-bin.zip

# Optional : install neo4j engine.
cd tinkergraph
./bin/gremlin-server.sh install org.apache.tinkerpop neo4j-gremlin 3.4.8
cd ..

php exakat.phar doctor
```

3.8 Installation guide with Composer

3.8.1 Composer installation first run

To install Exakat with composer, you can use the following commands:

```
mkdir exakat
cd exakat
composer require exakat/exakat
php vendor/bin/exakat install -v
```

The final command checks for the presence of Java and unZip utility. Then, it installs a local copy of a Gremlin server. This is needed to run Exakat.

To run your first audit, use the following commands:

```
php vendor/bin/exakat init -p sculpin -R 'https://github.com/sculpin/sculpin.git'
php vendor/bin/exakat project -p sculpin
```

The final audit is now in the *projects/sculpin/report* directory.

3.9 Using multiple PHP versions

You need at least one version of PHP to run exakat. This version needs the curl, hash, tokenizer, hash and sqlite3 extensions. They all are part of the core.

Extra PHP-CLI versions allow more linting of the code. They only need to have the tokenizer extension available.

Exakat recommends PHP 7.4.4 (or newer version) to run Exakat. We also recommend the installation of PHP versions 5.6, 7.1, 7.2, 7.3, 7.4 and 8.0 (aka php-src master).

To install easily various versions of PHP, use the ondrej repository. Check The main PPA for PHP (7.4, 7.3, 7.2, 7.1, 7.0, 5.6). You may also check the dotdeb repository, at dotdeb instruction or compile PHP yourself.

3.10 Installation guide with Docker

There are multiple ways to use exakat with docker. There is an image with a full exakat installation, which run with a traditional installation, or inside the audited code. Or, You may use Docker with a standard installation, to run useful part, such as a specific PHP version or the central database.

image:: images/exakat-and-docker.png

3.10.1 Docker image for Exakat with projects folder

Installation with Docker is easy, and convenient. It hides the dependency of the graph database, and keeps all files in the 'projects' folder, created in the working directory.

Currently, Docker installation only ships with one PHP version (7.3), and with support for bazaar, composer, git, mercurial, svn, and zip.

- Install Docker
- Start Docker

- Pull exakat. The official docker page is exakat/exakat.

```
docker pull exakat/exakat
```

- Check-run exakat :

```
docker run -it -v $(pwd)/projects:/usr/src/exakat/projects --rm --name my-exakat␣
↪exakat/exakat exakat version
docker run -it -v $(pwd)/projects:/usr/src/exakat/projects --rm --name my-exakat␣
↪exakat/exakat exakat doctor
```

- Init a project :

```
docker run -it -v $(pwd)/projects:/usr/src/exakat/projects --rm --name my-exakat␣
↪exakat/exakat exakat init -p <project name> -R <vcs_url>
```

- Run exakat :

```
docker run -it -v $(pwd)/projects:/usr/src/exakat/projects --rm --name my-exakat␣
↪exakat/exakat exakat project -p <project name>
```

- Run exakat directly in the code base. For that, the code needs to have the .exakat.yml or .exakat.ini file available at the root. Then, you may call exakat with the 'project' command, without other options.

```
docker run -it -v $(pwd)/projects:/usr/src/exakat/projects --rm --name my-exakat␣
↪exakat/exakat exakat project
```

For large code bases, it may be necessary to increase the allocated memory for the graph database. Do this by using the JAVA_OPTIONS environment variable when you start the docker command : this example gives 2Gb of RAM to the graphdb. That should cover medium size applications.

```
docker run -it -e JAVA_OPTIONS="-Xms32m -Xmx2g" -v $(pwd)/projects:/usr/src/exakat/
↪projects --rm --name my-exakat exakat/exakat exakat
```

You may run any exakat command by prefixing it with the following command :

```
docker run -it -v $(pwd)/projects:/usr/src/exakat/projects --rm --name my-exakat␣
↪exakat/exakat exakat
```

You may also create a handy shortcut, by creating an exakat.sh script and put it in your PATH :

```
cat 'docker run -it -v $(pwd)/projects:/usr/src/exakat/projects --rm --name my-exakat␣
↪exakat/exakat exakat $1' > /etc/local/sbin/exakat.sh
chmod u+x  /etc/local/sbin/exakat.sh
./exakat.sh version
```

3.10.2 Docker image for Exakat with projects folder

To run exakat inside the audited code, you must configure the *.exakat.ini* or *.exakat.yaml* file. See Add Exakat To Your CI Pipeline.

Then, you can run the following command, with docker :

```
docker run -it --rm -v `$pwd`:/src exakat/exakat:latest exakat project -v
```

3.10.3 Docker PHP image with Exakat

Exakat recognizes docker images configured as PHP binaries. Instead of configuring exakat with local binaries, such as */usr/bin/php*, you may configure a specific PHP version with a docker image.

Open the *config/exakat.ini* file, at the root of the exakat installation, and use the following value :

```
// configuration with the 'tetraweb/php:5.5' image.
;php55 = tetraweb/php:5.5
php56 = tetraweb/php:5.6
# classic configuration with local binary
php73 = /usr/bin/php
```

The image may be any docker image that provides a PHP binary. We suggest using tetraweb/php, which supports PHP 5.5 to 7.1. There are other images available, and you may also roll out your own.

3.10.4 Docker Gremlin image with Exakat

Exakat is able to use only the central database, Gremlin, as a docker image. This is convenient, as the database is only a temporary database, and those data are not necessary for producing the final reports.

This image is under construction, and will be soon available.

3.11 Installation guide as Github Action

3.11.1 Github Action

Github Action is a way to “Automate, customize, and execute your software development workflows right in your repository”. Exakat may be run on Github platform.

3.11.2 Github Action for Exakat

To add Exakat to your repository on Github, create a file *.github/workflows/test.yml*, at the root of your repository (*.github/workflows* might already exists).

In the file, use the following YAML code. It will create an automatic action, on push and pull_request actions, that runs Exakat and display the issues found in the workflow panel. It is also possible to run manually this action.

```
on: [push, pull_request]
name: Test
jobs:
  exakat:
    name: Exakat
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Exakat
      uses: docker://exakat/exakat-ga
```

Note : it is recommended to edit this file directly on github.com, as it cannot be pushed from a remote repository.

Then, you can use the *Action* button, next to ‘Pull requests’.

3.11.3 Exakat Docker image for Github Action

A Docker image is released with Exakat's version automatically, to be used with Github Action. It is available at https://hub.docker.com/r/exakat/exakat-ga.

You can run it in any given directory like this:

```
cd /path/to/code
docker pull exakat/exakat-ga
docker run --rm -it -v ${PWD}:/app exakat/exakat-ga:latest
```

3.11.4 Installation guide for optional tools

Exakat is able to use a variety of tools to access PHP code to audit. Some external tools are necessary. You can check which tools are recognized locally with the *exakat doctor -v* command.

- Bazaar : the *bzr* command must be available.
- composer : the *composer* command must be available.
- CVS : the *cvs* command must be available
- Git : the *git* command must be available.
- mercurial : the *hg* must be available
- Svn : the *svn* command must be available.
- tgz : the *tar* and *gunzip* commands must be available
- tbz : the *tar* and *bunzip2* commands must be available.
- rar : the *rar* commands must be available.
- zip : the *unzip* command must be available.
- 7z : the *7z* command must be available

The binaries above are used with the *init* and *update* commands, to get the source code. They are optional.

CHAPTER 4

Upgrading

4.1 Upgrading

Upgrade exakat with the *upgrade* command.

php exakat.phar upgrade

Exakat returns the current status :

This needs some updating (Current : 0.9.7c, Latest: 1.2.6)

To make exakat update itself, runs the same command, with the *-u* option. Exakat will then download the file, check the sums, and replace itself.

4.2 Upgrading manually

Exakat is a PHP phar archive. Download the latest version from dist.exakat.io and replace it.

4.3 Upgrading gremlin-server

Exakat installs the last version of gremlin at installation time. Usually, there is no need to upgrade the database when upgrading : changing the phar file is sufficient.

However, to enjoy the new features, or keep up to date, it is recommended to upgrade the gremlin server.

To upgrade gremlin-server, remove the old 'tinkergraph' folder from your installation. If exakat was installed following the installation instruction, this folder is located next to *exakat.phar*.

Then, run again the installation instruction, only for gremlin.

CHAPTER 5

Tutorials

Here are four tutorials to run Exakat on your code. You may install exakat with the projects folder, and centralize your audits in one place, or run exakat in-code, right from the source code. You may also run exakat with a bare-metal installation, or as a docker container.

- Bare metal install
- with projects folder
- within the code
- Docker container
- with projects folder
- within the code

All four tutorials offer the same steps : + Project initialisation + Audit run + Reports access

5.1 Bare metal install, with projects folder

5.1.1 Installation

Refer to the _Installation section to install Exakat.

5.1.2 Initialization

First, fetch the code to be audited. This has to be done once.

```
php exakat.phar init -p sculpin -R https://github.com/sculpin/sculpin
```

This command inits the project in the 'projects' folder, with the name 'sculpin', then clone the code with the provided repository.

Exakat requires a copy of the code. When accessing via VCS, such as git, mercurial, svn, etc., read-only access is sufficient and recommended. Exakat doesn't write anything in the code.

More information on _Commands.

5.1.3 Execution

After initialization, an audit may be run :

```
php exakat.phar project -p sculpin
```

This command runs the whole cycle : code loading, code audits and report building. It works without initial configuration.

Once it is finished, the reports are in the folder *projects/sculpin/report* (HTML version). Simply open the 'projects/sculpin/report/index.html' file in a browser.

5.1.4 More reports

Once the 'project' command has been fully run, you may run the 'report' command to access different report. Usually, 'Ambassador' has the most complete report, but other focused reports are available.

It is possible to access all report, even if another project is being processed.

```
php exakat.phar report -p sculpin -format Uml -file uml
```

This export the current project in UML format. The file is called 'uml.dot' : dot is added by exakat, as the report has to be opened by graphviz compatible software.

The full list of available reports are in the 'Command' section.

Once it is finished, the reports are in the folder *projects/sculpin/**.

5.1.5 New run

After some modification in the code, commit them in the repository. Then, run :

```
php exakat.phar update  -p sculpin
php exakat.phar project -p sculpin
```

This update the repository to the last modification, then runs the whole analysis. If the code is not using a VCS repository, such as git, mercurial, SVN, etc. Then the update command has no impact on the code. You should update the code manually, by replacing it with a newer version.

Once it is finished, the report are in the same previous folders : *projects/sculpin/report* (HTML version).

The reports replace any previous report. To keep a report of a previous version, move it away from the current location, and give it another name.

5.2 Bare metal install, within the code

This tutorial runs exakat from the source code repository.

5.2.1 Installation

Refer to the _Installation section to install Exakat.

5.2.2 Initialization

Go to the directory that contains the source code.

Create a configuration file called *.exakat.yml*, with the following content :

```
project: "name"
```

This is the minimum configuration for that file. You may read more about _Configuration in the dedicated section.

5.2.3 Execution

After creating the configuration file above, an audit may be run :

```
docker run -it --rm -w /src -v $(pwd):/src --entrypoint "/usr/src/exakat/exakat.phar"␣
↪exakat/exakat:latest project
```

This command runs the whole cycle : code loading, code audits and report building. It works without initial configuration.

Once it is finished, the reports are in the current folder. Simply open the 'report/index.html' file in a browser.

5.2.4 More reports

When running exakat inside code, audits must be configured before the run of the audit.

Edit the .exakat.yml file, and add the following lines :

```
project: "name"
project_reports:
  - Uml
  - Plantuml
  - Ambassador
```

Then, run the audit as explained in the previous section.

This configuration produces 3 reports : "Ambassador", which is the default report, "Uml", available in the 'uml.dot' file, and "Plantuml", that may be opened with plantuml.

The full list of available reports are in the 'Command' section.

5.2.5 New run

After some modification in the code, run again exakat with the same command than the first time. Since the audit is run within the code source, no update operation is needed.

Check the *config.ini* file before running the audit, to check if all the reports you want are configureds.

```
docker run -it --rm -w /src -v $(pwd):/src --entrypoint "/usr/src/exakat/exakat.phar"␣
↪exakat/exakat:latest project
```

5.3 Docker container, within the code folder

This tutorial runs exakat audits from the source code repository, with a docker container.

5.3.1 Installation

Refer to the _Installation section to install Exakat on docker.

5.3.2 Initialization

Go to the directory that contains the source code.

Create a configuration file called *.exakat.yml*, with the following content :

```
project: "name"
```

This is the minimum configuration for that file. You may read more about _Configuration in the dedicated section.

5.3.3 Execution

After creating the configuration file, an audit may be run from the same directory:

```
docker run -it --rm -v $(`pwd`):/src exakat/exakat:latest exakat project
```

This command runs the whole cycle : code loading, code audits and report building. It works without initial configuration.

Once it is finished, the report is displayed on the standard output (aka, the screen).

5.3.4 More reports

When running exakat inside code, reports must be configured before the run of the audit : they will be build immediately.

Edit the .exakat.yml file, and add the following lines :

```
project: "name"
project_reports:
  - Uml
  - Plantuml
  - Ambassador
```

Then, run the audit as explained in the previous section.

This configuration produces 3 reports : "Ambassador", which is the default report, "Uml", available in the 'uml.dot' file, and "Plantuml", that may be opened with plantuml.

The full list of available reports are in the _Reports section.

5.3.5 New run

After adding some modifications to the code, run again exakat with the same command than the first time. Since the audit is run within the code source, no explicit update operation is needed.

Check the *.exakat.yml* file before running the audit, to check if all the reports you want are configured.

```
docker run -it --rm -w /src -v $(pwd):/src --entrypoint "/usr/src/exakat/exakat.phar"␣
↪exakat/exakat:latest project
```

5.4 Docker container, with projects folder

This tutorial runs exakat audits, when source code are organized in the *projects* folder. Any folder will do, since exakat is now hosted in the docker image.

5.4.1 Initialization

Go to the directory that contains the ‘projects’ folder.

Init the project with the following command :

```
docker run -it --rm -v /Users/famille/Desktop/analyzeG3/projects:/usr/src/exakat/
↪projects exakat/exakat:latest exakat init -p sculpin -R https://github.com/sculpin/
↪sculpin -git
```

This will create a ‘projects/sculpin’ folder, with various documents and folder. The most important folder being ‘code’, where the code of the project is fetched, an cached. See _Commands for more details about the *init* command.

5.4.2 Execution

After creating the project, an audit may be run from the same directory:

```
docker run -it --rm -v /Users/famille/Desktop/analyzeG3/projects:/usr/src/exakat/
↪projects exakat/exakat:dev exakat project -p sculpin
```

This command runs the whole cycle : code loading, code audits and report building.

Once it is finished, the report is available in the *projects/sculpin/report/* folder. Open *projects/sculpin/report/index.htmll* with a browser.

5.4.3 More reports

When running exakat with the projects folder, reports may be configured before the run of the audit, in the config.ini file, or in command line, or extracted after the run.

After a first audit, use the *report* command. Here is an example with the *Uml* report.

```
docker run -it --rm -v /Users/famille/Desktop/analyzeG3/projects:/usr/src/exakat/
↪projects exakat/exakat:dev exakat report -p sculpin -format Uml
```

Reports may only be build if the analysis they depend on, were already processed.

In command line, use the *-format* option, multiple times if necessary.

```
docker run -it --rm -v /Users/famille/Desktop/analyzeG3/projects:/usr/src/exakat/
↪projects exakat/exakat:dev exakat project -p sculpin -format Uml
```

In config.ini, edit the *projects/sculpin/report/config.ini* file, and add the following lines :

```
project_reports[] = 'Uml';
project_reports[] = 'Plantuml';
project_reports[] = 'Ambassador';
```

Then, run the audit as explained in the previous section.

The full list of available reports are in the _Reports section.

5.4.4 New run

After adding some modifications to the code and committing them, you need to update the code before running it again : otherwise, it will run on the previous version of the code.

```
docker run -it --rm -v /Users/famille/Desktop/analyzeG3/projects:/usr/src/exakat/
↪projects exakat/exakat:dev exakat update -p sculpin
docker run -it --rm -v /Users/famille/Desktop/analyzeG3/projects:/usr/src/exakat/
↪projects exakat/exakat:dev exakat project -p sculpin
```

CHAPTER 6

Frequently Asked Questions

6.1 Summary

6.2 I need special command to get my code

If Exakat has no documented method to reach your code, you may use this process :

```
php exakat.phar init -p <your project name>
cd ./projects/<your project name>
mkdir code
// here, do whatever it takes to put all your code in 'code' folder
cd -
php exakat.phar project -p <your project name>
```

Send a message on Github.com/exakat/exakat to mention your specific method.

6.3 Can I checkout that branch?

Currently (Version 0.12.2), there is no way to request a tag or a branche or a revision when cloning the code.

The best way is to reach the 'code' folder, and make the change there. Unless with 'init' or 'update', exakat doesn't make any change to the code.

```
php exakat.phar init -p myProject -R url://my/git/repository
cd ./projects/myProject/code
git branch notMasterBranch
cd -
php exakat.phar project -p myProject
```

6.4 Can I clone with my ssh keys?

When using git, or any vcs, the current shell user's SSH keys may be used to access the repository. When using a remote installation, or a docker image, the keys won't be accessible.

The fallback solution is to init an empty project, clone the code from the Shell (with the keys), and then run project.

```
php exakat.phar init -p myProject
cd ./projects/myProject
git clone url://myprivate/git/repository code
cd -
php exakat.phar project -p myProject
```

6.5 After init, my project has no code!

Check in the projects/<name>/config.ini file : if values were provided, you'll find them there.

In case the code was not found during init, then do the following :

:: cd projects/<name>/ git clone ssh://project/URL code cd - php exakat.phar files -p <name>

If you're using some other method than git, then just collect the code in a 'code' folder in the <name> project and run the 'files' command.

6.6 The project is too big

There is a soft limit in config/exakat.ini, called 'token_limit' that initially prevents analysis of projects over 1 million tokens. That's roughly 125k LOC, more than most code source.

If you need to run exakat on larger sources, you may change this value to make it as large as possible. Then, the physical capacities of the machine, specially RAM, will be the actual limit.

It may be interesting to 'ignore_dir[]', from projects/<name>/config.ini.

6.7 Java Out Of Memory Error

By default, java is allowed to run with 512mb of RAM. That may be too little for the code being studied.

Set the environment variable $JAVA_OPTIONS to give larger quantities of RAM. For example : 'export JAVA_OPTIONS='-Xms1024m -Xmx6096m'; or 'setenv JAVA_OPTIONS='-Xms1024m -Xmx6096m'

Xms is the memory allocation at start, and Xmx is the maximum allocation. With some experimentation, 6G handles the largest

6.8 How can I run a very large project?

Here are a few steps you can try when running exakat on a very large project.

- Update project/<name>/config.ini, and use ignore_dirs[] and include_dirs[] to exclude as much code as possible. Notably, frameworks, data in PHP files, tests, cache, translations, etc.
- Set environment variable $JAVA_OPTIONS to large quantities of RAM : JAVA_OPTIONS='-Xms1024m -Xmx6096m';
- Check that your installation is running with 'gsneo4j' and not 'tinkergraph', in config/exakat.ini.

6.9 Does exakat runs on Java 8?

Exakat itself runs with PHP 7.0+. Exakat runs with a gremlin database : gremlin-server 3.2.x is supported, which runs on Java 8.

Java 9 is experimental, and is being tested. Java 7 used to be working, but is not supported anymore : it may still work, though.

6.10 Where can I find the report

Reports are available after running at least the following commands :

```
php exakat.phar init -p <your project name> -R <code source repo>
php exakat.phar project -p <your project name>
```

The default report is the HTML report, called 'Ambassador'. You'll find it in ./projects/<your project name>/report.

Other reports, build with 'report' command, will also be saved there, with different names.

6.11 Can I run exakat on local code?

There are several ways to do that : use symbolic links, make a copy of the source.

```
php exakat.phar init -p <your project name> -R <path/to/the/code> -symlink
php exakat.phar init -p <your project name> -R <path/to/the/code> -copy
php exakat.phar init -p <your project name> -R <path/to/the/code> -git
```

Symlink will branch exakat directly into the code; -copy makes a copy of the code (this means the code will never be updated without manual intervention); git (or other vcs) may also be used with local repositories.

Exakat do not modify any existing source code : it only access it for reading purpose, then works on a separated database. As a defensive security measure, we suggest that exakat should work on a read-only copy of the code.

6.12 Can I ignore a dir or a file?

Yes. After initing a project, open the projects/<project name>/config.ini file, and update the ignore_dir line. For example, to ignore a behat test folder, and to ignore any file called ‘license’ :

```
ignore_dirs[] = '/behat/';
ignore_dirs[] = 'license';
```

You may also include files, by using the include_dir[] line. Including files is processed after ignoring them, so you may include files in folders that were previously ignored.

6.13 *Can I audit only one folder in vendor?*

You can use ignore_dirs to exclude everything in the source tree, then use include_dirs to include specific folders.

:: # exclude everything ignore_dirs[] = ‘/’;

> # include intended folder include_dirs[] = ‘/vendor/exakat’;

6.14 Can I run Exakat with PHP 5?

It is recommended to run exakat with PHP 7.0 and more recent. Older version are not so well tested, since they have reached their end of life.

Note that you may test your code on PHP 5.x, while running Exakat on PHP 7.0. There are 2 distinct configuration options in Exakat. ‘php’ is the path to the PHP binary that runs Exakat : this one should be PHP 7.0+. ‘phpxx’ are the path to the PHP helpers, that are used to tokenized and lint the target PHP code. This is where PHP 5.x may be configured.

```
; where and which PHP executable are available
php   = /usr/local/sbin/php71

php52 =
php53 = /usr/local/sbin/php53
php54 =
php55 =
php56 =
```

(continues on next page)

(continued from previous page)

```
php70 =
php71 =
php72 =
php73 =
```

Above is an example of a exakat configuration file, where Exakat is run with PHP 7.1 and process code with PHP 5.3.

6.15 I get the error 'The executable 'ansible-playbook' Vagrant is trying to run was not found'

This error is displayed when the host machine doesn't have Ansible installed. Install ansible, and try again to provision.

6.16 Can I run exakat on Windows?

Currently, Windows is not supported, though it might be some day.

Until then, you may run Exakat with Vagrant, or with Docker.

6.17 Does exakat send my code to a central server?

When run from the sources, Exakat has everything it needs to fulfill its mission. There is no central server that does the job, and requires the transmission of the code.

When running an audit on the Saas service of Exakat, the code is processed on our servers.

6.18 "cat: write error: Broken pipe" : is it bad?

Exakat currently runs some piped commands, with xargs so as to make some operations parallel. When the following command ends up before the reading all the data from the first command, such a warning is emitted.

It has no impact on exakat's processing of the code.

See also cat: write error: Broken pipe.

CHAPTER 7

Exakat commands

7.1 List of commands :

- *anonymize*
- *baseline*
- *catalog*
- *clean*
- *cleandb*
- *doctor*
- *help*
- *init*
- *project*
- *report*
- *remove*
- *show*
- *update*
- *upgrade*
- *install*

7.2 anonymize

Read files, directory or projects, and produce a anonymized version of the code. Consistence between variables and names is preserved ($a is always replaced with the same name). PHP language structures, such as eval, isset or unset are preserved, though other native functions are not.

File structure is not preserved : all files are renamed, and the hiearchy is flattented in one folder. As such, code is probably un-runnable if it relies on inclusions.

Non-PHP files, non-lintable or files that produces one PHP token are ignored.

7.2.1 Command

```
exakat anonymize -p <project>
exakat anonymize -d <directory>
exakat anonymize -file <filename>
```

7.2.2 Options

Option	Req	Description
-p	No	Project name. Should be filesystem compatible (avoid /, : or) This takes into account <project> configuration
-d	No	Directory to anonymize. Results aree in <directory>.anon
-file	No	File to anonymize. Results are in <file>.anon
-v	No	Verbose mode

7.2.3 Tips

- -*R* is not compulsory : you may omit it, then, provide PHP files in the *projects/<name>/code* folder by the mean you want.

:: _baseline:

7.3 baseline

Baseline manage previous audits that may be used as a baseline for new audits.

A Baseline is a previous audit, that has already reviewed the code. It has identified issues and code. Later, after some code modification, a new audit is run. When we want to know the new issues, or the removed ones, it has to be compared to a baseline.

This is a help command, to help find the available values for various options.

7.3.1 Commands

Command	Description
list	List all available baselines. Default action
remove	Removes a baseline, using its name or its auto-id
save	Save the current audit, when it exists, as the last base, with the provided name.

:: _catalog:

7.4 catalog

Catalog list all available rulesets and reports with the current exakat.

This is a help command, to help find the available values for various options.

7.4.1 Options

Option	Req	Description
-json	No	Returns the catalog as JSON, for further processing.

:: _clean:

7.5 clean

Cleans the provided project of everything except the config.ini and the code.

This is a maintenance command, that removes all produced files and folder, and restores a project to its initial state.

7.5.1 Options

Option	Req	Description
-p	Yes	Project name. Should be an existing project.
-v	No	Verbose mode

:: _cleandb:

7.6 cleandb

Cleans the graph database.

This is a maintenance command, that removes all produced data and scripts, and restores the exakat database to its empty state.

By default, the database is cleaned with graph commands, letting the server do the cleaning.

The -Q option makes the same cleaning with a full restart of the server. This is cleaner, and faster if the database was big or in some instable state.

7.6.1 Options

Option	Req	Description
-Q	No	Cleans the database by restarting it, and removing files.
-stop	No	Stops gremlin server
-start	No	Starts gremlin server, without removing files.
-restart	No	Restarts gremlin server, without removing files.
-v	No	Verbose mode

:: _doctor:

7.7 doctor

Check the current installation of Exakat.

7.7.1 Command

```
exakat doctor
```

7.7.2 Results

```
PHP :
    version                : 7.0.1
    curl                   : Yes
    sqlite3                : Yes
    tokenizer              : Yes

java :
    installed              : Yes
    type                   : Java(TM) SE Runtime Environment (build 1.8.0_40-b25)
    version                : 1.8.0_40
    $JAVA_HOME             : /Library/Java/JavaVirtualMachines/jdk1.8.0_40.jdk/Contents/
 ↪Home

neo4j :
    version                : Neo4j 2.2.6
    port                   : 7474
    authentication         : Not enabled (Please, enable it)
    gremlinPlugin          : Configured.
    gremlinJar             : neo4j/plugins/gremlin-plugin/gremlin-java-2.7.0-SNAPSHOT.
 ↪jar
    scriptFolder           : Yes
    pid                    : 20895
    running                : Yes
    running here           : Yes
    gremlin                : Yes
    $NEO4J_HOME            : /Users/famille/Desktop/analyze/neo4j

folders :
    config-folder          : Yes
    config.ini             : Yes
    projects folder        : Yes
    progress               : Yes
    in                     : Yes
    out                    : Yes
    projects/test          : Yes
    projects/default       : Yes
    projects/onepage       : Yes

PHP 5.2 :
    configured             : No
```

(continues on next page)

(continued from previous page)

```
PHP 5.3 :
    configured             : Yes
    installed              : Yes
    version                : 5.3.29
    short_open_tags        : Off
    timezone               : Europe/Amsterdam
    tokenizer              : Yes
    memory_limit           : -1

PHP 5.4 :
    configured             : Yes
    installed              : Yes
    version                : 5.4.45
    short_open_tags        : Off
    timezone               : Europe/Amsterdam
    tokenizer              : Yes
    memory_limit           : 384M

PHP 5.5 :
    configured             : Yes
    installed              : Yes
    version                : 5.5.30
    short_open_tags        : Off
    timezone               : Europe/Amsterdam
    tokenizer              : Yes
    memory_limit           : -1

PHP 5.6 :
    configured             : /usr/local/sbin/php56
    installed              : Yes
    version                : 5.6.16
    short_open_tags        : Off
    timezone               : Europe/Amsterdam
    tokenizer              : Yes
    memory_limit           : -1

PHP 7.0 :
    configured             : Yes
    version                : 7.0.1
    short_open_tags        : Off
    timezone               :
    tokenizer              : Yes
    memory_limit           : -1

PHP 7.1 :
    configured             : Yes
    version                : 7.1.0-dev
    short_open_tags        : Off
    timezone               :
    tokenizer              : Yes
    memory_limit           : 128M

git :
    installed              : Yes
    version                : 2.7.0
```

(continues on next page)

(continued from previous page)

```
hg :
    installed            : Yes
    version              : 3.6.3

svn :
    installed            : Yes
    version              : 1.9.3

bzr :
    installed            : No
    optional             : Yes

composer :
    installed            : Yes
    version              : 1.0.0-alpha11

wget :
    installed            : Yes
    version              : GNU Wget 1.17.1 built on darwin15.2.0.

zip :
    installed            : Yes
    version              : 3.0
```

# Tips

- The *PHP* section is the PHP binary used to run Exakat.
- The *PHP x.y* sections are the PHP binaries used to check the code.
- Optional installations (such as svn, zip, etc.) are not necessarily reported if they are not installed.

7.7.3 Options

Option	Req	Description
-p	No	Displays the project-specific configuration. Otherwise, only displays general configuration.
-json	No	Displays the project-specific configuration in json format, to stdout
-v	No	Verbose mode : include helpers configurations
-q	No	Quiet mode : runs doctor, and install checks, but displays nothing. This is useful to automate installation finalization

:: _help:

7.8 help

Displays the help section.

```
php exakat.phar help
```

7.8.1 Results

This displays :

```
[Usage] :   php exakat.phar init -p <Project name> -R <Repository>
            php exakat.phar project -p <Project name>
            php exakat.phar doctor
            php exakat.phar version
```

:: _init:

7.9 init

Initialize a new project.

7.9.1 Command

```
exakat init -p <project> [-R vcs_url] [-git|-svn|-bzr|-hg|-composer|-symlink|-copy|-
↪tgz|-7z|-zip] [-v] [-D]
```

7.9.2 Options

Option	Req	Description
-p	Yes	Project name. Should be filesystem compatible (avoid /, : or)
-R	No	URL to the VCS repository. Anything compatible with the expected VCS.
-git	No	Use git client (also, default value if no clue is given in the VCS URL)
-svn	No	Use SVN client
-bzr	No	Use Bazar client
-hg	No	Use Mercurial (hg) client
-composer	No	Use Composer client
-symlink	No	-R path is symlinked. Directory is never accessed for writing.
-copy	No	-R path is recursively copied.
-zip	No	-R is a ZIP archive, local or remote
-tgz	No	-R is a .tar.gzip archive, local or remote
-tbz	No	-R is a .tar.bz2 archive, local or remote
-rar	No	-R is a .rar archive, local or remote
-7z	No	-R is a .7z archive, local or remote
-v	No	Verbose mode
-D	No	First erase any pre-existing project with the same name

7.9.3 Tips

- *-R* is not compulsory : you may omit it, then, provide PHP files in the *projects/<name>/code* folder by the mean you want.
- Default VCS used is git.
- *-D* removes any previous project before doing the init.

- Archives (zip, tar.gz, tar.bz, 7z, rar, etc.) depends on external tools to unpack them. They depends on PHP to reach the file, locally or remotely.

7.9.4 Examples

```
# Clone Exakat with Git
php exakat.phar init -p exakat -R https://github.com/exakat/exakat.git

# Download Spip with Zip
php exakat init -p spip2 -zip -R http://files.spip.org/spip/stable/spip-3.1.zip

# Download PHPMyadmin,
php exakat.phar init -p pma2 -tgz -R https://files.phpmyadmin.net/phpMyAdmin/4.6.4/
↪phpMyAdmin-4.6.4-all-languages.tar.gz

# Make a local copy of PHPMyadmin,
php exakat.phar init -p copyProject -copy -R projects/phpmyadmin/code/

# Make a local symlink with the local webserver,
php exakat.phar init -p symlinkProject -symlink -R /var/www/public_html
```

:: _project:

7.10 project

Runs a new analyze on a project.

The results of the analysis are available in the *projects/<name>/* folder. *report* and *faceted* are two HTML reports.

7.10.1 Command

```
exakat project -p <project> [-v]
```

7.10.2 Options

Option	Req	Description
-p	Yes	Project name. Should be filesystem compatible (avoid /, : or)
-v	No	Verbose mode

:: _remove:

7.11 remove

Destroy a project. All code source, configuration and any results from exakat are destroyed.

7.11.1 Command

```
exakat remove -p <project> [-v]
```

7.11.2 Options

Option	Req	Description
-p	Yes	Project name. Should be filesystem compatible (avoid /, : or)
-v	No	Verbose mode

:: _remove:

7.12 show

Displays the the full command line to create an exakat project.

7.12.1 Command

```
exakat show -p <project>
```

7.12.2 Options

Option	Req	Description
-p	Yes	Project name. Should be filesystem compatible (avoid /, : or)

:: _report:

7.13 report

Produce a report for a project.

Reports may be produced as soon as exakat has reach the phase of 'analysis'. If the analysis phase hasn't finished, then some results may be unavailable. Run report again later to get the full report. For example, the 'Uml' report may be run fully as soon as exakat is in analysis phase.

It is possible to extract a report even after the graph database has been cleaned. This allows running several projects one after each other, yet have access to several reports.

7.13.1 Command

```
exakat report -p <project> -format <Format> [-file <file>] [-v]
```

7.13.2 Options

Option	Req	Description
-p	Yes	Project name. Should be filesystem compatible (avoid /, : or)
-v	No	Verbose mode
-format	No	Which format to extract. Available formats : Devoops, Faceted, FacetedJson, Json, OnepageJson, Text, Uml, Xml Default is 'Text'
-file	No	File or directory name for the report. Adapted file extension is added. Report is located in the projects/<project>/ folder Default is 'stdout', but varies with format.
-T	No	Ruleset's results. All the analyses in this ruleset are reported. Note that the report format may override this configuration : for example Ambassador manage its own list of analyses. Uses this with Text format. Has priority over the -P option
-P	No	Analyzer's results. Only one analysis's is reported. Note that the report format may override this configuration : for example Ambassador manage its own list of analyses. Uses this with Text format. Has lower priority than the -T option

7.13.3 Report formats

All reports are detailed in the ref:*Reports <reports>* section.

Report	Description
Amabassador	HTML format, with all available reports in one compact format.
Devoops	HTML format, deprecated.
Json	JSON format.
Text	Text format. One issue per line, with description, file, line.
Codesniffer	Text format, similar to Codesniffer report style.
Uml	Dot format. All classes/interfaces/traits hierarchies, and grouped by name- spaces.
Xml	XML format.
All	All availble format, using default naming

:: _update:

7.14 update

Update the code base of a project.

7.14.1 Command

```
exakat update -p <project> [-v]
```

7.14.2 Options

Option	Req	Description
-p	Yes	Project name. Should be filesystem compatible (avoid /, : or)
-v	No	Verbose mode

:: _upgrade:

7.15 upgrade

Upgrade exakat itself. By default, this command only checks for the availability of a new version : it doesn't upgrade immediately.

Use -u option to actually replace the current phar archive.

Use -version option to downgrade or upgrade to a specific version.

In case the upgrade command file, you may also download manually the *.phar* from the exakat.io website : www.exakat.io. Then replace the current version with the new one.

7.15.1 Command

```
exakat upgrade
```

7.15.2 Options

Option	Req	Description
-u	Yes	Actually upgrades exakat. Without it, it is a dry run.
-version	No	Select a specific Exakat version and update to it. By default, it upgrades to the latest version, as published on the https://www.exakat.io/ site. Example value : 1.8.8

7.16 Install

Install exakat's graph dependency. This command is an integrated installation script, and it is only accessible once the .phar is downloaded locally.

7.16.1 Command

```
mkdir exakat
cd exakat

// Download exakat.phar, like this, or any other valid means
curl -o exakat.phar https://www.exakat.io/versions/index.php?file=latest
exakat.phar upgrade
```

7.16.2 Options

Option	Req	Description
-u	Yes	Actually upgrades exakat. Without it, it is a dry run.
-version	No	Select a specific Exakat version and update to it. By default, it upgrades to the latest version, as published on the https://www.exakat.io/ site. Example value : 1.8.8

CHAPTER 8

Rulesets

8.1 Presentation

Analysis are grouped in different rulesets, that may be run independantly. Each ruleset has a focus target,

Rulesets runs all its analysis and any needed dependency.

Rulesets are configured with the -T option, when running exakat in command line. For example :

```
php exakat.phar analyze -p <project> -T <Security>
```

8.2 List of rulesets

Here is the list of the current rulesets supported by Exakat Engine.

Name	Description
Analyze	Check for common best practices.
CI-checks	Quick check for common best practices.
Dead code	Check the unused code or unreachable code.
Suggestions	List of possible modernisation of the PHP code.
Compatibility-PHP74	List features that are incompatible with PHP 7.4. It is known as php-src, work in progress.
Compatibility-PHP73	List features that are incompatible with PHP 7.3.
Compatibility-PHP72	List features that are incompatible with PHP 7.2.
Compatibility-PHP71	List features that are incompatible with PHP 7.1.
Compatibility-PHP80	Work in progress. The first rules are in, but far from finished
Performances	Check the code for slow code.
Security	Check the code for common security bad practices, especially in the Web environnement.
Top10	The most common issues found in the code
ClassReview	A set of rules dedicate to class hygiene
LintButWontExec	Check the code for common errors that will lead to a Fatal error on production, but lint fine.
Compatibility-PHP70	List features that are incompatible with PHP 7.0.
Compatibility-PHP56	List features that are incompatible with PHP 5.6.
Compatibility-PHP55	List features that are incompatible with PHP 5.5.
Compatibility-PHP54	List features that are incompatible with PHP 5.4.
Compatibility-PHP53	List features that are incompatible with PHP 5.3.
Coding Conventions	List coding conventions violations.
Semantics	Checks the meanings found the names of the code.
Typechecks	Checks related to types.
Rector	Suggests configuration to apply changes with Rector
php-cs-fixable	Suggests configuration to apply changes with PHP-CS-FIXER

Note : in command line, don't forget to add quotes to rulesets' names that include white space.

8.3 Rulesets details

8.3.1 Analyze

This ruleset centralizes a large number of classic trap and pitfalls when writing PHP.

Total : 412 analysis

- *$this Belongs To Classes Or Traits*
- *$this Is Not An Array*
- *$this Is Not For Static Methods*

- *@ Operator*
- *Abstract Or Implements*
- *Abstract Static Methods*
- *Access Protected Structures*
- *Accessing Private*
- *Adding Zero*
- *Aliases Usage*
- *Already Parents Interface*
- *Already Parents Trait*
- *Altering Foreach Without Reference*
- *Alternative Syntax Consistence*
- *Always Positive Comparison*
- *Ambiguous Array Index*
- *Ambiguous Static*
- *Ambiguous Visibilities*
- *Array_Fill() With Objects*
- *Array_merge Needs Array Of Arrays*
- *Assert Function Is Reserved*
- *Assign And Compare*
- *Assign Default To Properties*
- *Assign With And*
- *Assigned Twice*
- *Assumptions*
- *Avoid Optional Properties*
- *Avoid Parenthesis*
- *Avoid Substr() One*
- *Avoid Using stdClass*
- *Avoid get_class()*
- *Avoid mb_dectect_encoding()*
- *Avoid option arrays in constructors*
- *Bad Constants Names*
- *Bail Out Early*
- *Break Outside Loop*
- *Buried Assignation*
- *Callback Needs Return*
- *Can't Extend Final*

- *Can’t Throw Throwable*
- *Cant Implement Traversable*
- *Cant Instantiate Class*
- *Cast To Boolean*
- *Casting Ternary*
- *Catch Overwrite Variable*
- *Catch Undefined Variable*
- *Check All Types*
- *Check JSON*
- *Check On __Call Usage*
- *Class Could Be Final*
- *Class Should Be Final By Ocramius*
- *Class Without Parent*
- *Class, Interface Or Trait With Identical Names*
- *Clone With Non-Object*
- *Coalesce And Concat*
- *Common Alternatives*
- *Compared Comparison*
- *Concat And Addition*
- *Concat Empty String*
- *Concrete Visibility*
- *Constant Class*
- *Constant Comparison*
- *Constants Created Outside Its Namespace*
- *Constants With Strange Names*
- *Continue Is For Loop*
- *Could Be Abstract Class*
- *Could Be Else*
- *Could Be Static*
- *Could Be Stringable*
- *Could Make A Function*
- *Could Use Short Assignation*
- *Could Use __DIR__*
- *Could Use self*
- *Could Use str_repeat()*
- *Crc32() Might Be Negative*

- *Cyclic References*
- *Dangling Array References*
- *Deep Definitions*
- *Dependant Abstract Classes*
- *Dependant Trait*
- *Deprecated Functions*
- *Different Argument Counts*
- *Don't Change Incomings*
- *Don't Echo Error*
- *Don't Pollute Global Space*
- *Don't Read And Write In One Expression*
- *Don't Send $this In Constructor*
- *Don't Unset Properties*
- *Dont Change The Blind Var*
- *Dont Collect Void*
- *Dont Mix ++*
- *Double Assignation*
- *Double Instructions*
- *Double Object Assignation*
- *Drop Else After Return*
- *Echo With Concat*
- *Else If Versus Elseif*
- *Empty Blocks*
- *Empty Classes*
- *Empty Function*
- *Empty Instructions*
- *Empty Interfaces*
- *Empty List*
- *Empty Namespace*
- *Empty Traits*
- *Empty Try Catch*
- *Eval() Usage*
- *Exit() Usage*
- *Failed Substr Comparison*
- *Fn Argument Variable Confusion*
- *Foreach On Object*

- *Foreach Reference Is Not Modified*
- *Forgotten Interface*
- *Forgotten Thrown*
- *Forgotten Visibility*
- *Forgotten Whitespace*
- *Fully Qualified Constants*
- *Global Usage*
- *Hardcoded Passwords*
- *Hash Algorithms*
- *Hidden Nullable*
- *Hidden Use Expression*
- *Htmlentities Calls*
- *Identical Conditions*
- *Identical Consecutive Expression*
- *Identical On Both Sides*
- *If With Same Conditions*
- *Iffectations*
- *Illegal Name For Method*
- *Implement Is For Interface*
- *Implemented Methods Are Public*
- *Implied If*
- *Implode() Arguments Order*
- *Inclusion Wrong Case*
- *Incompatible Signature Methods*
- *Incompatible Signature Methods With Covariance*
- *Incompilable Files*
- *Inconsistent Elseif*
- *Indices Are Int Or String*
- *Infinite Recursion*
- *Instantiating Abstract Class*
- *Insufficient Typehint*
- *Interfaces Don't Ensure Properties*
- *Interfaces Is Not Implemented*
- *Invalid Constant Name*
- *Invalid Pack Format*
- *Invalid Regex*

- *Is Actually Zero*
- *Is_A() With String*
- *Logical Mistakes*
- *Logical Should Use Symbolic Operators*
- *Logical To in_array*
- *Lone Blocks*
- *Long Arguments*
- *Lost References*
- *Make Global A Property*
- *Max Level Of Nesting*
- *Mbstring Third Arg*
- *Mbstring Unknown Encoding*
- *Memoize MagicCall*
- *Merge If Then*
- *Method Collision Traits*
- *Method Could Be Static*
- *Method Signature Must Be Compatible*
- *Methods Without Return*
- *Mismatch Parameter And Type*
- *Mismatch Parameter Name*
- *Mismatch Properties Typehints*
- *Mismatch Type And Default*
- *Mismatched Default Arguments*
- *Mismatched Ternary Alternatives*
- *Mismatched Typehint*
- *Missing Abstract Method*
- *Missing Cases In Switch*
- *Missing Include*
- *Missing New ?*
- *Missing Parenthesis*
- *Missing Returntype In Method*
- *Mixed Concat And Interpolation*
- *Modernize Empty With Expression*
- *Modified Typed Parameter*
- *Multiple Alias Definitions*
- *Multiple Alias Definitions Per File*

- *Multiple Class Declarations*
- *Multiple Constant Definition*
- *Multiple Declaration Of Strict_types*
- *Multiple Identical Trait Or Interface*
- *Multiple Index Definition*
- *Multiple Type Variable*
- *Multiples Identical Case*
- *Multiply By One*
- *Must Call Parent Constructor*
- *Must Return Methods*
- *Negative Power*
- *Nested Ifthen*
- *Nested Ternary*
- *Never Used Parameter*
- *Never Used Properties*
- *Next Month Trap*
- *No Append On Source*
- *No Boolean As Default*
- *No Choice*
- *No Class In Global*
- *No Direct Call To Magic Method*
- *No Direct Usage*
- *No Empty Regex*
- *No Hardcoded Hash*
- *No Hardcoded Ip*
- *No Hardcoded Path*
- *No Hardcoded Port*
- *No Literal For Reference*
- *No Magic With Array*
- *No Need For Else*
- *No Need For Triple Equal*
- *No Parenthesis For Language Construct*
- *No Public Access*
- *No Real Comparison*
- *No Reference For Ternary*
- *No Reference On Left Side*

- *No Return Used*
- *No Self Referencing Constant*
- *No Spread For Hash*
- *No array_merge() In Loops*
- *No get_class() With Null*
- *No isset() With empty()*
- *Non Ascii Variables*
- *Non Nullable Getters*
- *Non Static Methods Called In A Static*
- *Non-constant Index In Array*
- *Not Equal Is Not !==*
- *Not Not*
- *Null Or Boolean Arrays*
- *Objects Don't Need References*
- *Old Style Constructor*
- *Old Style __autoload()*
- *One Variable String*
- *Only Variable For Reference*
- *Only Variable Passed By Reference*
- *Only Variable Returned By Reference*
- *Or Die*
- *Overwritten Exceptions*
- *Overwritten Literals*
- *Overwritten Source And Value*
- *PHP Keywords As Names*
- *Parent First*
- *Parent, Static Or Self Outside Class*
- *Pathinfo() Returns May Vary*
- *Possible Infinite Loop*
- *Possible Missing Subpattern*
- *Pre-increment*
- *Preprocessable*
- *Print And Die*
- *Printf Number Of Arguments*
- *Property Could Be Local*
- *Property Used In One Method Only*

- *Queries In Loops*
- *Randomly Sorted Arrays*
- *Redeclared PHP Functions*
- *Redefined Class Constants*
- *Redefined Default*
- *Redefined Private Property*
- *Relay Function*
- *Repeated Interface*
- *Repeated Regex*
- *Repeated print()*
- *Results May Be Missing*
- *Return True False*
- *Same Conditions In Condition*
- *Same Variable Foreach*
- *Scalar Are Not Arrays*
- *Scalar Or Object Property*
- *Several Instructions On The Same Line*
- *Short Open Tags*
- *Should Chain Exception*
- *Should Make Alias*
- *Should Make Ternary*
- *Should Typecast*
- *Should Use Coalesce*
- *Should Use Constants*
- *Should Use Explode Args*
- *Should Use Local Class*
- *Should Use Prepared Statement*
- *Should Use SetCookie()*
- *Should Yield With Key*
- *Silently Cast Integer*
- *Static Loop*
- *Static Methods Called From Object*
- *Static Methods Can't Contain $this*
- *Strange Name For Constants*
- *Strange Name For Variables*
- *Strict Comparison With Booleans*

- *String May Hold A Variable*
- *Strings With Strange Space*
- *Strpos()-like Comparison*
- *Strtr Arguments*
- *Suspicious Comparison*
- *Swapped Arguments*
- *Switch To Switch*
- *Switch Without Default*
- *Ternary In Concat*
- *Test Then Cast*
- *Throw Functioncall*
- *Throw In Destruct*
- *Throws An Assignement*
- *Timestamp Difference*
- *Too Many Array Dimensions*
- *Too Many Dereferencing*
- *Too Many Finds*
- *Too Many Injections*
- *Too Many Local Variables*
- *Too Many Native Calls*
- *Trait Not Found*
- *Typehint Must Be Returned*
- *Typehinted References*
- *Uncaught Exceptions*
- *Unchecked Resources*
- *Unconditional Break In Loop*
- *Undefined Class Constants*
- *Undefined Classes*
- *Undefined Constant Name*
- *Undefined Constants*
- *Undefined Functions*
- *Undefined Insteadof*
- *Undefined Interfaces*
- *Undefined Parent*
- *Undefined Properties*
- *Undefined Trait*

- *Undefined Variable*
- *Undefined ::class*
- *Undefined static:: Or self::*
- *Unknown Parameter Name*
- *Unknown Pcre2 Option*
- *Unkown Regex Options*
- *Unpreprocessed Values*
- *Unresolved Classes*
- *Unresolved Instanceof*
- *Unresolved Use*
- *Unset In Foreach*
- *Unsupported Types With Operators*
- *Unthrown Exception*
- *Unused Arguments*
- *Unused Class Constant*
- *Unused Classes*
- *Unused Global*
- *Unused Inherited Variable In Closure*
- *Unused Returned Value*
- *Use === null*
- *Use Class Operator*
- *Use Constant*
- *Use Constant As Arguments*
- *Use Instanceof*
- *Use Named Boolean In Argument Definition*
- *Use PHP Object API*
- *Use Pathinfo*
- *Use Positive Condition*
- *Use System Tmp*
- *Use With Fully Qualified Name*
- *Use array_slice()*
- *Use const*
- *Use random_int()*
- *Used Once Property*
- *Used Once Variables (In Scope)*
- *Used Once Variables*

- *Useless Abstract Class*
- *Useless Alias*
- *Useless Brackets*
- *Useless Casting*
- *Useless Catch*
- *Useless Check*
- *Useless Constructor*
- *Useless Final*
- *Useless Global*
- *Useless Instructions*
- *Useless Interfaces*
- *Useless Parenthesis*
- *Useless Referenced Argument*
- *Useless Return*
- *Useless Switch*
- *Useless Unset*
- *Uses Default Values*
- *Using $this Outside A Class*
- *Using Deprecated Method*
- *Var Keyword*
- *Variable Is Not A Condition*
- *Weak Typing*
- *While(List() = Each())*
- *Written Only Variables*
- *Wrong Access Style to Property*
- *Wrong Argument Type*
- *Wrong Number Of Arguments*
- *Wrong Optional Parameter*
- *Wrong Parameter Type*
- *Wrong Range Check*
- *Wrong Returned Type*
- *Wrong Type For Native PHP Function*
- *Wrong Type With Call*
- *Wrong Typed Property Default*
- *Wrong fopen() Mode*
- *__DIR__ Then Slash*

- *__toString() Throws Exception*
- *array_key_exists() Works On Arrays*
- *array_merge() And Variadic*
- *error_reporting() With Integers*
- *eval() Without Try*
- *func_get_arg() Modified*
- *include_once() Usage*
- *list() May Omit Variables*
- *preg_replace With Option e*
- *self, parent, static Outside Class*
- *strip_tags Skips Closed Tag*
- *strpos() Too Much*
- *var_dump()... Usage*

8.3.2 CI-checks

This ruleset is a collection of important rules to run in a CI pipeline.

Total : 177 analysis

- *@ Operator*
- *Adding Zero*
- *Aliases Usage*
- *Altering Foreach Without Reference*
- *Always Positive Comparison*
- *Assign And Compare*
- *Assign With And*
- *Avoid Parenthesis*
- *Avoid Substr() One*
- *Avoid get_class()*
- *Callback Needs Return*
- *Cant Implement Traversable*
- *Casting Ternary*
- *Check JSON*
- *Check On __Call Usage*
- *Class Without Parent*
- *Coalesce And Concat*
- *Concat And Addition*
- *Constant Class*

- *Constants With Strange Names*
- *Could Use Short Assignation*
- *Could Use __DIR__*
- *Could Use str_repeat()*
- *Dangling Array References*
- *Deprecated Functions*
- *Don't Echo Error*
- *Don't Unset Properties*
- *Drop Else After Return*
- *Else If Versus Elseif*
- *Empty Blocks*
- *Empty Namespace*
- *Exit() Usage*
- *Failed Substr Comparison*
- *Foreach Reference Is Not Modified*
- *Forgotten Visibility*
- *Forgotten Whitespace*
- *Hidden Use Expression*
- *Htmlentities Calls*
- *Identical Conditions*
- *Identical On Both Sides*
- *If With Same Conditions*
- *Implied If*
- *Implode() Arguments Order*
- *Indices Are Int Or String*
- *Interfaces Is Not Implemented*
- *Invalid Pack Format*
- *Invalid Regex*
- *Is Actually Zero*
- *Is_A() With String*
- *Logical Mistakes*
- *Logical Should Use Symbolic Operators*
- *Lone Blocks*
- *Mbstring Third Arg*
- *Mbstring Unknown Encoding*
- *Merge If Then*

- *Missing Parenthesis*
- *Missing Returntype In Method*
- *Multiple Alias Definitions*
- *Multiple Alias Definitions Per File*
- *Multiple Class Declarations*
- *Multiple Constant Definition*
- *Multiple Identical Trait Or Interface*
- *Multiple Index Definition*
- *Multiples Identical Case*
- *Multiply By One*
- *Must Return Methods*
- *Negative Power*
- *Nested Ternary*
- *Next Month Trap*
- *No Choice*
- *No Class In Global*
- *No Direct Call To Magic Method*
- *No Empty Regex*
- *No Literal For Reference*
- *No Magic With Array*
- *No Parenthesis For Language Construct*
- *No Real Comparison*
- *No Reference For Ternary*
- *No Reference On Left Side*
- *No array_merge() In Loops*
- *No isset() With empty()*
- *Non Static Methods Called In A Static*
- *Not Equal Is Not !==*
- *Not Not*
- *Objects Don't Need References*
- *One Variable String*
- *Or Die*
- *Overwritten Exceptions*
- *Possible Missing Subpattern*
- *Pre-increment*
- *Print And Die*

- *Printf Number Of Arguments*
- *Redeclared PHP Functions*
- *Redefined Class Constants*
- *Redefined Default*
- *Repeated Regex*
- *Repeated print()*
- *Results May Be Missing*
- *Return True False*
- *Same Conditions In Condition*
- *Same Variable Foreach*
- *Scalar Are Not Arrays*
- *Should Chain Exception*
- *Should Make Alias*
- *Should Make Ternary*
- *Should Typecast*
- *Should Use Coalesce*
- *Should Use Explode Args*
- *Should Use Prepared Statement*
- *Should Yield With Key*
- *Silently Cast Integer*
- *Static Methods Called From Object*
- *Static Methods Can't Contain $this*
- *Strict Comparison With Booleans*
- *Strings With Strange Space*
- *Strpos()-like Comparison*
- *Strtr Arguments*
- *Switch Without Default*
- *Ternary In Concat*
- *Throw Functioncall*
- *Throw In Destruct*
- *Throws An Assignement*
- *Timestamp Difference*
- *Typehint Must Be Returned*
- *Typehinted References*
- *Unchecked Resources*
- *Unconditional Break In Loop*

- *Undefined Class Constants*
- *Undefined Constants*
- *Undefined Functions*
- *Undefined Insteadof*
- *Undefined Interfaces*
- *Undefined Properties*
- *Undefined Trait*
- *Undefined Variable*
- *Undefined ::class*
- *Unknown Parameter Name*
- *Unused Inherited Variable In Closure*
- *Use === null*
- *Use Class Operator*
- *Use Constant*
- *Use Constant As Arguments*
- *Use Instanceof*
- *Use PHP Object API*
- *Use Pathinfo*
- *Use System Tmp*
- *Use array_slice()*
- *Use const*
- *Use random_int()*
- *Useless Alias*
- *Useless Brackets*
- *Useless Casting*
- *Useless Catch*
- *Useless Check*
- *Useless Final*
- *Useless Instructions*
- *Useless Parenthesis*
- *Useless Unset*
- *Uses Default Values*
- *While(List() = Each())*
- *Wrong Access Style to Property*
- *Wrong Number Of Arguments*
- *Wrong Optional Parameter*

- *Wrong Parameter Type*
- *Wrong Returned Type*
- *Wrong Type For Native PHP Function*
- *Wrong Type With Call*
- *Wrong Typed Property Default*
- *Wrong fopen() Mode*
- *__DIR__ Then Slash*
- *error_reporting() With Integers*
- *eval() Without Try*
- *list() May Omit Variables*
- *preg_replace With Option e*
- *strip_tags Skips Closed Tag*
- *strpos() Too Much*
- *var_dump()... Usage*

8.3.3 ClassReview

This ruleset focuses on classes construction issues, and their related structures : traits, interfaces, methods, properties, constants.

Total : 51 analysis

- *Avoid Self In Interface*
- *Avoid option arrays in constructors*
- *Cancel Common Method*
- *Class Could Be Final*
- *Class Without Parent*
- *Classes Mutually Extending Each Other*
- *Could Be Abstract Class*
- *Could Be Class Constant*
- *Could Be Parent Method*
- *Could Be Private Class Constant*
- *Could Be Protected Class Constant*
- *Could Be Protected Method*
- *Could Be Protected Property*
- *Could Be Static*
- *Could Use self*
- *Cyclic References*
- *Dependant Abstract Classes*

- *Different Argument Counts*
- *Disconnected Classes*
- *Double Object Assignation*
- *Exceeding Typehint*
- *Final Class Usage*
- *Final Methods Usage*
- *Fossilized Method*
- *Hidden Nullable*
- *Insufficient Property Typehint*
- *Interfaces Don't Ensure Properties*
- *Interfaces Is Not Implemented*
- *Memoize MagicCall*
- *Method Could Be Private Method*
- *Method Could Be Static*
- *Mismatch Properties Typehints*
- *Missing Abstract Method*
- *Modified Typed Parameter*
- *No Self Referencing Constant*
- *Non Nullable Getters*
- *Nullable Without Check*
- *Property Could Be Local*
- *Property Could Be Private Property*
- *Raised Access Level*
- *Redefined Property*
- *Self Using Trait*
- *Uninitilized Property*
- *Unreachable Class Constant*
- *Unused Class Constant*
- *Unused Trait In Class*
- *Useless Interfaces*
- *Useless Typehint*
- *Wrong Access Style to Property*
- *Wrong Returned Type*
- *Wrong Typed Property Default*

8.3.4 Coding Conventions

This ruleset centralizes all analysis related to coding conventions. Sometimes, those are easy to extract with static analysis, and so here they are. No all o them are available.

Total : 27 analysis

- *All Uppercase Variables*
- *Bracketless Blocks*
- *Close Tags*
- *Constant Comparison*
- *Don't Be Too Manual*
- *Echo Or Print*
- *Empty Slots In Arrays*
- *Heredoc Delimiter*
- *Interpolation*
- *Mistaken Concatenation*
- *Mixed Concat And Interpolation*
- *Multiple Classes In One File*
- *No Plus One*
- *Non-lowercase Keywords*
- *One Letter Functions*
- *Order Of Declaration*
- *Return With Parenthesis*
- *Should Be Single Quote*
- *Similar Integers*
- *Unusual Case For PHP Functions*
- *Use With Fully Qualified Name*
- *Use const*
- *Wrong Case Namespaces*
- *Wrong Class Name Case*
- *Wrong Function Name Case*
- *Wrong Typehinted Name*
- *Yoda Comparison*

8.3.5 CompatibilityPHP53

This ruleset centralizes all analysis for the migration from PHP 5.2 to 5.3.

Total : 79 analysis

- *Anonymous Classes*

- *Binary Glossary*
- *Break With 0*
- *Cant Inherit Abstract Method*
- *Cant Use Return Value In Write Context*
- *Child Class Removes Typehint*
- *Class Const With Array*
- *Closure May Use $this*
- *Coalesce Equal*
- *Concat And Addition*
- *Const Visibility Usage*
- *Const With Array*
- *Constant Scalar Expressions*
- *Continue Is For Loop*
- *Define With Array*
- *Dereferencing String And Arrays*
- *Direct Call To __clone()*
- *Ellipsis Usage*
- *Exponent Usage*
- *Flexible Heredoc*
- *Foreach With list()*
- *Function Subscripting*
- *Generator Cannot Return*
- *Group Use Declaration*
- *Group Use Trailing Comma*
- *Hash Algorithms Incompatible With PHP 5.3*
- *Hash Algorithms Incompatible With PHP 7.1-*
- *Integer As Property*
- *List Short Syntax*
- *List With Keys*
- *List With Reference*
- *Malformed Octal*
- *Methodcall On New*
- *Mixed Keys Arrays*
- *Multiple Definition Of The Same Argument*
- *Multiple Exceptions Catch()*
- *New Functions In PHP 5.4*

- *New Functions In PHP 5.5*
- *New Functions In PHP 5.6*
- *New Functions In PHP 7.0*
- *New Functions In PHP 7.3*
- *No List With String*
- *No Reference For Static Property*
- *No Return For Generator*
- *No String With Append*
- *No Substr Minus One*
- *No get_class() With Null*
- *Non Static Methods Called In A Static*
- *Null On New*
- *PHP 7.0 New Classes*
- *PHP 7.0 New Interfaces*
- *PHP 7.0 Scalar Typehints*
- *PHP 7.1 Scalar Typehints*
- *PHP 7.2 Scalar Typehints*
- *PHP 7.3 Last Empty Argument*
- *PHP5 Indirect Variable Expression*
- *PHP7 Dirname*
- *Parenthesis As Parameter*
- *Php 7 Indirect Expression*
- *Php 7.1 New Class*
- *Php 7.2 New Class*
- *Php7 Relaxed Keyword*
- *Short Syntax For Arrays*
- *Switch With Too Many Default*
- *Trailing Comma In Calls*
- *Typed Property Usage*
- *Unicode Escape Partial*
- *Unicode Escape Syntax*
- *Unpacking Inside Arrays*
- *Use Const And Functions*
- *Use Lower Case For Parent, Static And Self*
- *Use Nullable Type*
- *Variable Global*

- *::class*
- *__debugInfo() Usage*
- *ext/dba*
- *ext/fdf*
- *ext/ming*
- *isset() With Constant*

8.3.6 CompatibilityPHP54

This ruleset centralizes all analysis for the migration from PHP 5.3 to 5.4.

Total : 75 analysis

- *Anonymous Classes*
- *Break With Non Integer*
- *Calltime Pass By Reference*
- *Cant Inherit Abstract Method*
- *Cant Use Return Value In Write Context*
- *Child Class Removes Typehint*
- *Class Const With Array*
- *Coalesce Equal*
- *Concat And Addition*
- *Const Visibility Usage*
- *Const With Array*
- *Constant Scalar Expressions*
- *Continue Is For Loop*
- *Define With Array*
- *Dereferencing String And Arrays*
- *Direct Call To __clone()*
- *Ellipsis Usage*
- *Exponent Usage*
- *Flexible Heredoc*
- *Foreach With list()*
- *Functions Removed In PHP 5.4*
- *Generator Cannot Return*
- *Group Use Declaration*
- *Group Use Trailing Comma*
- *Hash Algorithms Incompatible With PHP 5.3*
- *Hash Algorithms Incompatible With PHP 5.4/5.5*

- *Hash Algorithms Incompatible With PHP 7.1-*
- *Integer As Property*
- *List Short Syntax*
- *List With Keys*
- *List With Reference*
- *Malformed Octal*
- *Mixed Keys Arrays*
- *Multiple Definition Of The Same Argument*
- *Multiple Exceptions Catch()*
- *New Functions In PHP 5.5*
- *New Functions In PHP 5.6*
- *New Functions In PHP 7.0*
- *New Functions In PHP 7.3*
- *No List With String*
- *No Reference For Static Property*
- *No Return For Generator*
- *No String With Append*
- *No Substr Minus One*
- *No get_class() With Null*
- *Non Static Methods Called In A Static*
- *Null On New*
- *PHP 7.0 New Classes*
- *PHP 7.0 New Interfaces*
- *PHP 7.0 Scalar Typehints*
- *PHP 7.1 Scalar Typehints*
- *PHP 7.2 Scalar Typehints*
- *PHP 7.3 Last Empty Argument*
- *PHP5 Indirect Variable Expression*
- *PHP7 Dirname*
- *Parenthesis As Parameter*
- *Php 7 Indirect Expression*
- *Php 7.1 New Class*
- *Php 7.2 New Class*
- *Php7 Relaxed Keyword*
- *Switch With Too Many Default*
- *Trailing Comma In Calls*

- *Typed Property Usage*
- *Unicode Escape Partial*
- *Unicode Escape Syntax*
- *Unpacking Inside Arrays*
- *Use Const And Functions*
- *Use Lower Case For Parent, Static And Self*
- *Use Nullable Type*
- *Variable Global*
- *::class*
- *__debugInfo() Usage*
- *crypt() Without Salt*
- *ext/mhash*
- *isset() With Constant*

8.3.7 CompatibilityPHP55

This ruleset centralizes all analysis for the migration from PHP 5.4 to 5.5.

Total : 67 analysis

- *Anonymous Classes*
- *Cant Inherit Abstract Method*
- *Child Class Removes Typehint*
- *Class Const With Array*
- *Coalesce Equal*
- *Concat And Addition*
- *Const Visibility Usage*
- *Const With Array*
- *Constant Scalar Expressions*
- *Continue Is For Loop*
- *Define With Array*
- *Direct Call To __clone()*
- *Ellipsis Usage*
- *Exponent Usage*
- *Flexible Heredoc*
- *Functions Removed In PHP 5.5*
- *Generator Cannot Return*
- *Group Use Declaration*
- *Group Use Trailing Comma*

- *Hash Algorithms Incompatible With PHP 5.3*
- *Hash Algorithms Incompatible With PHP 5.4/5.5*
- *Hash Algorithms Incompatible With PHP 7.1-*
- *Integer As Property*
- *List Short Syntax*
- *List With Keys*
- *List With Reference*
- *Malformed Octal*
- *Multiple Definition Of The Same Argument*
- *Multiple Exceptions Catch()*
- *New Functions In PHP 5.6*
- *New Functions In PHP 7.0*
- *New Functions In PHP 7.3*
- *No List With String*
- *No Reference For Static Property*
- *No Return For Generator*
- *No String With Append*
- *No Substr Minus One*
- *No get_class() With Null*
- *Non Static Methods Called In A Static*
- *Null On New*
- *PHP 7.0 New Classes*
- *PHP 7.0 New Interfaces*
- *PHP 7.0 Scalar Typehints*
- *PHP 7.1 Scalar Typehints*
- *PHP 7.2 Scalar Typehints*
- *PHP 7.3 Last Empty Argument*
- *PHP5 Indirect Variable Expression*
- *PHP7 Dirname*
- *Parenthesis As Parameter*
- *Php 7 Indirect Expression*
- *Php 7.1 New Class*
- *Php 7.2 New Class*
- *Php7 Relaxed Keyword*
- *Switch With Too Many Default*
- *Trailing Comma In Calls*

- *Typed Property Usage*
- *Unicode Escape Partial*
- *Unicode Escape Syntax*
- *Unpacking Inside Arrays*
- *Use Const And Functions*
- *Use Nullable Type*
- *Use password_hash()*
- *Variable Global*
- *__debugInfo() Usage*
- *ext/apc*
- *ext/mysql*
- *isset() With Constant*

8.3.8 CompatibilityPHP56

This ruleset centralizes all analysis for the migration from PHP 5.5 to 5.6.

Total : 57 analysis

- *$HTTP_RAW_POST_DATA Usage*
- *Anonymous Classes*
- *Cant Inherit Abstract Method*
- *Child Class Removes Typehint*
- *Coalesce Equal*
- *Concat And Addition*
- *Const Visibility Usage*
- *Continue Is For Loop*
- *Define With Array*
- *Direct Call To __clone()*
- *Flexible Heredoc*
- *Generator Cannot Return*
- *Group Use Declaration*
- *Group Use Trailing Comma*
- *Hash Algorithms Incompatible With PHP 5.3*
- *Hash Algorithms Incompatible With PHP 5.4/5.5*
- *Hash Algorithms Incompatible With PHP 7.1-*
- *Integer As Property*
- *List Short Syntax*
- *List With Keys*

- *List With Reference*
- *Malformed Octal*
- *Multiple Definition Of The Same Argument*
- *Multiple Exceptions Catch()*
- *New Functions In PHP 7.0*
- *New Functions In PHP 7.3*
- *No List With String*
- *No Reference For Static Property*
- *No Return For Generator*
- *No String With Append*
- *No Substr Minus One*
- *No get_class() With Null*
- *Non Static Methods Called In A Static*
- *Null On New*
- *PHP 7.0 New Classes*
- *PHP 7.0 New Interfaces*
- *PHP 7.0 Scalar Typehints*
- *PHP 7.1 Scalar Typehints*
- *PHP 7.2 Scalar Typehints*
- *PHP 7.3 Last Empty Argument*
- *PHP5 Indirect Variable Expression*
- *PHP7 Dirname*
- *Parenthesis As Parameter*
- *Php 7 Indirect Expression*
- *Php 7.1 New Class*
- *Php 7.2 New Class*
- *Php 8.0 Only TypeHints*
- *Php7 Relaxed Keyword*
- *Switch With Too Many Default*
- *Trailing Comma In Calls*
- *Typed Property Usage*
- *Unicode Escape Partial*
- *Unicode Escape Syntax*
- *Unpacking Inside Arrays*
- *Use Nullable Type*
- *Variable Global*

- *isset() With Constant*

8.3.9 CompatibilityPHP70

This ruleset centralizes all analysis for the migration from PHP 5.6 to 7.0.

Total : 49 analysis

- *Break Outside Loop*
- *Cant Inherit Abstract Method*
- *Child Class Removes Typehint*
- *Coalesce Equal*
- *Concat And Addition*
- *Const Visibility Usage*
- *Continue Is For Loop*
- *Empty List*
- *Flexible Heredoc*
- *Foreach Don't Change Pointer*
- *Group Use Trailing Comma*
- *Hash Algorithms Incompatible With PHP 5.3*
- *Hash Algorithms Incompatible With PHP 5.4/5.5*
- *Hash Algorithms Incompatible With PHP 7.1-*
- *Hexadecimal In String*
- *Integer As Property*
- *List Short Syntax*
- *List With Appends*
- *List With Keys*
- *List With Reference*
- *Magic Visibility*
- *Multiple Exceptions Catch()*
- *New Functions In PHP 7.3*
- *No Reference For Static Property*
- *No Substr Minus One*
- *No get_class() With Null*
- *PHP 7.0 Removed Directives*
- *PHP 7.0 Removed Functions*
- *PHP 7.1 Scalar Typehints*
- *PHP 7.2 Scalar Typehints*
- *PHP 7.3 Last Empty Argument*

- *Php 7 Indirect Expression*
- *Php 7.1 New Class*
- *Php 7.2 New Class*
- *Php 8.0 Only TypeHints*
- *Reserved Keywords In PHP 7*
- *Setlocale() Uses Constants*
- *Simple Global Variable*
- *Trailing Comma In Calls*
- *Typed Property Usage*
- *Union Typehint*
- *Unpacking Inside Arrays*
- *Use Nullable Type*
- *Usort Sorting In PHP 7.0*
- *ext/ereg*
- *func_get_arg() Modified*
- *mcrypt_create_iv() With Default Values*
- *preg_replace With Option e*
- *set_exception_handler() Warning*

8.3.10 CompatibilityPHP71

This ruleset centralizes all analysis for the migration from PHP 7.0 to 7.1.

Total : 36 analysis

- *Avoid Substr() One*
- *Cant Inherit Abstract Method*
- *Child Class Removes Typehint*
- *Coalesce Equal*
- *Concat And Addition*
- *Continue Is For Loop*
- *Flexible Heredoc*
- *Group Use Trailing Comma*
- *Hash Algorithms Incompatible With PHP 5.3*
- *Hash Algorithms Incompatible With PHP 5.4/5.5*
- *Hexadecimal In String*
- *Integer As Property*
- *Invalid Octal In String*
- *List With Reference*

- *New Functions In PHP 7.1*
- *New Functions In PHP 7.3*
- *No Reference For Static Property*
- *No get_class() With Null*
- *PHP 7.0 Removed Directives*
- *PHP 7.0 Removed Functions*
- *PHP 7.1 Microseconds*
- *PHP 7.1 Removed Directives*
- *PHP 7.2 Scalar Typehints*
- *PHP 7.3 Last Empty Argument*
- *Php 7.2 New Class*
- *Php 8.0 Only TypeHints*
- *Signature Trailing Comma*
- *String Initialization*
- *Trailing Comma In Calls*
- *Typed Property Usage*
- *Union Typehint*
- *Unpacking Inside Arrays*
- *Use random_int()*
- *Using $this Outside A Class*
- *ext/mcrypt*
- *preg_replace With Option e*

8.3.11 CompatibilityPHP72

This ruleset centralizes all analysis for the migration from PHP 7.1 to 7.2.

Total : 29 analysis

- *Avoid set_error_handler $context Argument*
- *Can't Count Non-Countable*
- *Coalesce Equal*
- *Concat And Addition*
- *Continue Is For Loop*
- *Flexible Heredoc*
- *Hash Algorithms Incompatible With PHP 5.3*
- *Hash Algorithms Incompatible With PHP 5.4/5.5*
- *Hash Will Use Objects*
- *List With Reference*

- *New Constants In PHP 7.2*
- *New Functions In PHP 7.2*
- *New Functions In PHP 7.3*
- *No Reference For Static Property*
- *No get_class() With Null*
- *PHP 7.2 Deprecations*
- *PHP 7.2 Object Keyword*
- *PHP 7.2 Removed Functions*
- *PHP 7.3 Last Empty Argument*
- *Php 7.2 New Class*
- *Php 8.0 Only TypeHints*
- *Signature Trailing Comma*
- *Throw Was An Expression*
- *Trailing Comma In Calls*
- *Typed Property Usage*
- *Undefined Constants*
- *Union Typehint*
- *Unpacking Inside Arrays*
- *preg_replace With Option e*

8.3.12 CompatibilityPHP73

This ruleset centralizes all analysis for the migration from PHP 7.2 to 7.3.

Total : 18 analysis

- *Assert Function Is Reserved*
- *Case Insensitive Constants*
- *Coalesce Equal*
- *Compact Inexistant Variable*
- *Concat And Addition*
- *Continue Is For Loop*
- *Don't Read And Write In One Expression*
- *New Functions In PHP 7.3*
- *Numeric Literal Separator*
- *PHP 7.3 Removed Functions*
- *PHP 74 New Directives*
- *Php 8.0 Only TypeHints*
- *Signature Trailing Comma*

- *Throw Was An Expression*
- *Typed Property Usage*
- *Union Typehint*
- *Unknown Pcre2 Option*
- *Unpacking Inside Arrays*

8.3.13 CompatibilityPHP74

This ruleset centralizes all analysis for the migration from PHP 7.3 to 7.4.

Total : 29 analysis

- *Concat And Addition*
- *Detect Current Class*
- *Don't Read And Write In One Expression*
- *Filter To add_slashes()*
- *Hash Algorithms Incompatible With PHP 7.4-*
- *Nested Ternary Without Parenthesis*
- *New Constants In PHP 7.4*
- *New Functions In PHP 7.4*
- *New Functions In PHP 8.0*
- *No More Curly Arrays*
- *PHP 7.4 Constant Deprecation*
- *PHP 7.4 Removed Directives*
- *PHP 7.4 Removed Functions*
- *PHP 7.4 Reserved Keyword*
- *Php 7.4 New Class*
- *Php 8.0 Only TypeHints*
- *Php 8.0 Variable Syntax Tweaks*
- *Php/UseMatch*
- *Reflection Export() Is Deprecated*
- *Scalar Are Not Arrays*
- *Signature Trailing Comma*
- *Throw Was An Expression*
- *Unbinding Closures*
- *Union Typehint*
- *array_key_exists() Works On Arrays*
- *curl_version() Has No Argument*
- *idn_to_ascii() New Default*

- *mb_strrpos() Third Argument*
- *openssl_random_pseudo_byte() Second Argument*

8.3.14 CompatibilityPHP80

This ruleset centralizes all analysis for the migration from PHP 7.4 to 8.0.

Total : 11 analysis

- *$php_errormsg Usage*
- *Cast Unset Usage*
- *Concat And Addition*
- *Mismatch Parameter Name*
- *Negative Start Index In Array*
- *Nullable With Constant*
- *Old Style Constructor*
- *PHP 8.0 Removed Constants*
- *PHP 8.0 Removed Directives*
- *PHP 8.0 Removed Functions*
- *Unsupported Types With Operators*

8.3.15 Dead code

This ruleset focuses on dead code : expressions or even structures that are written, valid but never used.

Total : 26 analysis

- *Can't Extend Final*
- *Empty Instructions*
- *Empty Namespace*
- *Exception Order*
- *Locally Unused Property*
- *Rethrown Exceptions*
- *Self Using Trait*
- *Undefined Caught Exceptions*
- *Unreachable Code*
- *Unresolved Catch*
- *Unresolved Instanceof*
- *Unset In Foreach*
- *Unthrown Exception*
- *Unused Classes*
- *Unused Constants*

- *Unused Functions*
- *Unused Inherited Variable In Closure*
- *Unused Interfaces*
- *Unused Label*
- *Unused Methods*
- *Unused Private Methods*
- *Unused Private Properties*
- *Unused Protected Methods*
- *Unused Returned Value*
- *Unused Use*
- *Useless Type Check*

8.3.16 LintButWontExec

This ruleset focuses on PHP code that lint (php -l), but that will not run. As such, this ruleset tries to go further than PHP, by connecting files, just like during execution.

Total : 29 analysis

- *Abstract Or Implements*
- *Can't Throw Throwable*
- *Cant Implement Traversable*
- *Classes Mutually Extending Each Other*
- *Clone With Non-Object*
- *Concrete Visibility*
- *Could Be Stringable*
- *Final Class Usage*
- *Final Methods Usage*
- *Incompatible Signature Methods*
- *Interfaces Is Not Implemented*
- *Method Collision Traits*
- *Method Signature Must Be Compatible*
- *Mismatch Properties Typehints*
- *Mismatch Type And Default*
- *Must Return Methods*
- *No Magic With Array*
- *No Self Referencing Constant*
- *Only Variable For Reference*
- *Raised Access Level*

- *Repeated Interface*
- *Trait Not Found*
- *Typehint Must Be Returned*
- *Undefined Insteadof*
- *Undefined Trait*
- *Useless Alias*
- *Using $this Outside A Class*
- *Wrong Typed Property Default*
- *self, parent, static Outside Class*

8.3.17 Performances

This ruleset focuses on performances issues : anything that slows the code's execution.

Total : 46 analysis

- *@ Operator*
- *Always Use Function With array_key_exists()*
- *Autoappend*
- *Avoid Concat In Loop*
- *Avoid Large Array Assignation*
- *Avoid Substr() One*
- *Avoid array_push()*
- *Avoid array_unique()*
- *Avoid glob() Usage*
- *Cache Variable Outside Loop*
- *Closure Could Be A Callback*
- *Could Use Short Assignation*
- *Do In Base*
- *Double array_flip()*
- *Echo With Concat*
- *Eval() Usage*
- *Fetch One Row Format*
- *For Using Functioncall*
- *Getting Last Element*
- *Global Inside Loop*
- *Isset() On The Whole Array*
- *Joining file()*
- *Make Magic Concrete*

- *Make One Call With Array*
- *No Count With 0*
- *No array_merge() In Loops*
- *No mb_substr In Loop*
- *Optimize Explode()*
- *Pre-increment*
- *Processing Collector*
- *Regex On Arrays*
- *Should Use Function*
- *Should Use array_column()*
- *Simple Switch*
- *Simplify Regex*
- *Slice Arrays First*
- *Slow Functions*
- *Substring First*
- *Use Class Operator*
- *Use PHP7 Encapsed Strings*
- *Use The Blind Var*
- *Use pathinfo() Arguments*
- *While(List() = Each())*
- *array_key_exists() Speedup*
- *fputcsv() In Loops*
- *time() Vs strtotime()*

8.3.18 Rector

RectorPHP is a reconstructor tool. It applies modifications in the PHP code automatically. Exakat finds results which may be automatically updated with rector.

Total : 3 analysis

- *Else If Versus Elseif*
- *Is_A() With String*
- *Preprocessable*

8.3.19 Security

This ruleset focuses on code security.

Total : 44 analysis

- *Always Anchor Regex*

- *Avoid Those Hash Functions*
- *Avoid sleep()/usleep()*
- *Check Crypto Key Length*
- *Compare Hash*
- *Configure Extract*
- *Direct Injection*
- *Don't Echo Error*
- *Dynamic Library Loading*
- *Encoded Simple Letters*
- *Eval() Usage*
- *Hardcoded Passwords*
- *Indirect Injection*
- *Integer Conversion*
- *Keep Files Access Restricted*
- *Minus One On Error*
- *Mkdir Default*
- *No ENT_IGNORE*
- *No Hardcoded Hash*
- *No Hardcoded Ip*
- *No Hardcoded Port*
- *No Net For Xml Load*
- *No Return Or Throw In Finally*
- *No Weak SSL Crypto*
- *Phpinfo*
- *Random Without Try*
- *Register Globals*
- *Safe Curl Options*
- *Safe HTTP Headers*
- *Session Lazy Write*
- *Set Cookie Safe Arguments*
- *Should Use Prepared Statement*
- *Should Use session_regenerateid()*
- *Sqlite3 Requires Single Quotes*
- *Switch Fallthrough*
- *Unserialize Second Arg*
- *Upload Filename Injection*

- *Use random_int()*
- *eval() Without Try*
- *filter_input() As A Source*
- *move_uploaded_file Instead Of copy*
- *parse_str() Warning*
- *preg_replace With Option e*
- *var_dump()... Usage*

8.3.20 Semantics

This ruleset focuses on human interpretation of the code. It reviews special values of literals, and named structures.

Total : 13 analysis

- *Class Function Confusion*
- *Duplicate Literal*
- *Fn Argument Variable Confusion*
- *Mismatch Parameter And Type*
- *One Letter Functions*
- *Parameter Hiding*
- *Prefix And Suffixes With Typehint*
- *Property Variable Confusion*
- *Semantic Typing*
- *Similar Integers*
- *Variables With One Letter Names*
- *Weird Array Index*
- *Wrong Typehinted Name*

8.3.21 Suggestions

This ruleset focuses on possibly better syntax than the one currently used. Those may be code modernization, alternatives, more efficient solutions, or simply left over from older versions.

Total : 92 analysis

- *** For Exponent*
- *Abstract Away*
- *Add Default Value*
- *Already Parents Interface*
- *Avoid Real*
- *Avoid Substr() One*
- *Cancel Common Method*

- *Closure Could Be A Callback*
- *Compact Inexistant Variable*
- *Complex Dynamic Names*
- *Could Be Constant*
- *Could Be Static Closure*
- *Could Make A Function*
- *Could Use Alias*
- *Could Use Compact*
- *Could Use Promoted Properties*
- *Could Use Try*
- *Could Use __DIR__*
- *Could Use array_fill_keys*
- *Could Use array_unique*
- *Could Use self*
- *Detect Current Class*
- *Directly Use File*
- *Don't Loop On Yield*
- *Dont Compare Typed Boolean*
- *Drop Else After Return*
- *Drop Substr Last Arg*
- *Echo With Concat*
- *Empty With Expression*
- *Function Subscripting, Old Style*
- *Implode One Arg*
- *Isset Multiple Arguments*
- *Isset() On The Whole Array*
- *Large Try Block*
- *Logical Should Use Symbolic Operators*
- *Mismatched Ternary Alternatives*
- *Multiple Unset()*
- *Multiple Usage Of Same Trait*
- *Named Regex*
- *Never Used Parameter*
- *No Need For get_class()*
- *No Parenthesis For Language Construct*
- *No Return Used*

- *One If Is Sufficient*
- *Overwritten Exceptions*
- *PHP7 Dirname*
- *Parent First*
- *Possible Alias Confusion*
- *Possible Increment*
- *Preprocess Arrays*
- *Randomly Sorted Arrays*
- *Repeated print()*
- *Return With Parenthesis*
- *Reuse Variable*
- *Set Aside Code*
- *Should Deep Clone*
- *Should Have Destructor*
- *Should Preprocess Chr()*
- *Should Use Coalesce*
- *Should Use Foreach*
- *Should Use Math*
- *Should Use Operator*
- *Should Use array_column()*
- *Should Use array_filter()*
- *Slice Arrays First*
- *Static Global Variables Confusion*
- *Strict Comparison With Booleans*
- *Substr To Trim*
- *Substring First*
- *Too Long A Block*
- *Too Many Children*
- *Too Many Parameters*
- *Too Much Indented*
- *Unitialized Properties*
- *Unreachable Code*
- *Unused Interfaces*
- *Use Array Functions*
- *Use Basename Suffix*
- *Use Case Value*

- *Use Count Recursive*
- *Use DateTimeImmutable Class*
- *Use List With Foreach*
- *Use Url Query Functions*
- *Use is_countable*
- *Use json_decode() Options*
- *Use session_start() Options*
- *Useless Default Argument*
- *Useless Typehint*
- *While(List() = Each())*
- *array_key_exists() Speedup*
- *list() May Omit Variables*
- *preg_match_all() Flag*

8.3.22 Top10

This ruleset is a selection of analysis, with the top 10 most common. Actually, it is a little larger than that.

Total : 28 analysis

- *Avoid Concat In Loop*
- *Avoid Real*
- *Avoid Substr() One*
- *Concat And Addition*
- *Could Use str_repeat()*
- *Dangling Array References*
- *Don't Unset Properties*
- *Failed Substr Comparison*
- *For Using Functioncall*
- *Logical Operators Favorite*
- *Logical Should Use Symbolic Operators*
- *Next Month Trap*
- *No Choice*
- *No Real Comparison*
- *No array_merge() In Loops*
- *Objects Don't Need References*
- *Possible Missing Subpattern*
- *Queries In Loops*
- *Repeated print()*

- *Should Yield With Key*
- *Strpos()-like Comparison*
- *Substring First*
- *Unitialized Properties*
- *Unresolved Instanceof*
- *Use List With Foreach*
- *Use const*
- *Used Once Variables*
- *fputcsv() In Loops*

8.3.23 Typechecks

This ruleset focuses on typehinting. Missing typehint, or inconsistent typehint, are reported.

Total : 23 analysis

- *Argument Should Be Typehinted*
- *Bad Typehint Relay*
- *Child Class Removes Typehint*
- *Could Be Callable*
- *Could Be Float*
- *Could Be Integer*
- *Could Be Iterable*
- *Could Be Null*
- *Could Be Parent*
- *Could Be Self*
- *Could Be String*
- *Could Be Void*
- *Fossilized Method*
- *Insufficient Typehint*
- *Mismatch Type And Default*
- *Mismatched Default Arguments*
- *Mismatched Typehint*
- *Missing Typehint*
- *No Class As Typehint*
- *Not A Scalar Type*
- *Useless Interfaces*
- *Wrong Argument Type*
- *Wrong Type With Call*

8.3.24 php-cs-fixable

[PHP-CS-fixer](https://github.com/FriendsOfPHP/PHP-CS-Fixer) is a tool to automatically fix PHP Coding Standards issues. It applies modifications in the PHP code automatically. Exakat finds results which may be automatically updated with php-cs-fixer.

Total : 11 analysis

- ** For Exponent
- Could Use __DIR__
- Don't Unset Properties
- Else If Versus Elseif
- Implode One Arg
- Isset Multiple Arguments
- Logical Should Use Symbolic Operators
- Multiple Unset()
- PHP7 Dirname
- Use === null
- Use Constant

CHAPTER 9

Rules list

9.1 Introduction

9.2 $HTTP_RAW_POST_DATA Usage

`$HTTP_RAW_POST_DATA` is deprecated, and should be replaced by `php://input`.

`$HTTP_RAW_POST_DATA` is deprecated since PHP 5.6.

It is possible to prepare code to this lack of feature by setting `always_populate_raw_post_data` to -1.

```
<?php

// PHP 5.5 and older
$postdata = $HTTP_RAW_POST_DATA;

// PHP 5.6 and more recent
$postdata = file_get_contents(php://input);

?>
```

See also $HTTP_RAW_POST_DATA variable.

9.2.1 Suggestions

- Use php://input with fopen() instead.

Short name	Php/RawPostDataUsage
Rulesets	*CompatibilityPHP56*
Severity	Major
Time To Fix	Slow (1 hour)

9.3 $php_errormsg Usage

$php_errormsg is removed since PHP 8.0. $php_errormsg tracks the last error message, with the directive *track_errors*. All was removed in PHP 8.0, and shall be replaced with error_get_last().

```
<?php

function foo() {
    global $php_errormsg;

    echo 'Last error: '.$php_errormsg;

    echo 'Also, last error: '.error_get_last();
}

?>
```

9.3.1 Suggestions

- Use error_get_last() instead.

Short name	Php/PhpErrorMsgUsage
Rulesets	*CompatibilityPHP80*
Php Version	8.0-
Severity	Minor
Time To Fix	Quick (30 mins)

9.4 $this Belongs To Classes Or Traits

$this variable represents the current object, inside a class or trait scope.

It is a pseudo-variable, and should be used within class's or trait's methods and not outside. It should also not be used in static methods.

PHP 7.1 is stricter and check for $this at several situations. Some are found by static analysis, some are dynamic analysis.

```
<?php

// as an argument
function foo($this) {
    // Using global
    global $this;
    // Using static (not a property)
    static $this;

    // Can't unset it
    unset($this);

    try {
        // inside a foreach
        foreach($a as $this) {   }
```

(continues on next page)

(continued from previous page)

```
        foreach($a as $this => $b) {   }
        foreach($a as $b => $this) {   }
    } catch (Exception $this) {
        // inside a catch
    }

    // with Variable Variable
    $a = this;
    $$a = 42;
}

class foo {
    function bar() {
        // Using references
        $a =& $this;
        $a = 42;

        // Using extract(), parse_str() or similar functions
        extract([this => 42]);  // throw new Error(Cannot re-assign $this)
        var_dump($this);
    }

    static function __call($name, $args) {
        // Using __call
        var_dump($this); // prints object(C)#1 (0) {}, php-7.0 printed NULL
        $this->test();   // prints ops
    }

}
?>
```

9.4.1 Suggestions

- Do not use *$this* as a variable name, except for the current object, in a class, trait or closure.

Short name	Classes/ThisIsForClasses
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*OpenEMR*

9.5 $this Is Not An Array

`$this` variable represents the current object and it is not an array.

This is unless the class (or its parents) has the `ArrayAccess` interface, or extends `ArrayObject` or `SimpleXMLElement`.

```
<?php

// $this is an array
class Foo extends ArrayAccess {
```

(continues on next page)

(continued from previous page)

```
    function bar() {
        ++$this[3];
    }
}

// $this is not an array
class Foo2 {
    function bar() {
        ++$this[3];
    }
}

?>
```

See also ArrayAccess, ArrayObject and The Basics.

9.5.1 Suggestions

- Extends `ArrayObject`, or a class that extends it, to use `$this` as an array too.
- Implements `ArrayAccess` to use `$this` as an array too.
- Use a property in the current class to store the data, instead of $this directly.

Short name	Classes/ThisIsNotAnArray
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)

9.6 $this Is Not For Static Methods

Static methods shouldn't use $this variable.

$this variable represents an object, the current object. It is not compatible with a static method, which may operate without any object.

While executing a static method, $this is actually set to NULL.

```
<?php

class foo {
    static $staticProperty = 1;

    // Static methods should use static properties
    static public function count() {
        return self::$staticProperty++;
    }

    // Static methods can't use $this
    static public function bar() {
        return $this->a;   // No $this usage in a static method
    }
}
```

(continues on next page)

(continued from previous page)

```

?>
```

See also Static Keyword <https://www.php.net/manual/en/language.oop5.'static.php>'_.

9.6.1 Suggestions

- Remove the static keyword on the method, and update all calls to this method to use $this
- Remove the usage of $this in the method, replacing it with static properties
- Make $this an argument (and change its name) : then, make the method a function

Short name	Classes/ThisIsNotForStatic
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	no-static-this

9.7 ** For Exponent

The operator `**` calculates exponents, also known as power.

Use it instead of the slower function pow(). This operator was introduced in PHP 5.6.

```
<?php
    $cube = pow(2, 3); // 8

    $cubeInPHP56 = 2 ** 3; // 8
?>
```

Be aware the the '-' operator has lower priority than the ** operator : this leads to the following confusing result.

```
<?php
    echo -3 ** 2;
    // displays -9, instead of 9
?>
```

This is due to the parser that processes separately − and the following number. Since `**` has priority, the power operation happens first.

Being an operator, `**` is faster than pow(). This is a microoptimisation.

See also Arithmetic Operators.

9.7.1 Suggestions

- Use the `**` operator
- For powers of 2, use the bitshift operators
- For literal powers of 2, consider using the `0xFFFFFFFFF` syntax.

Short name	Php/NewExponent
Rulesets	*Suggestions*, *php-cs-fixable*
Php Version	With PHP 5.6 and more recent
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high
Examples	*Traq*, *TeamPass*

9.8 ::class

PHP has a special class constant to hold the name of the class : `class` keyword. It represents the class name that is used in the left part of the operator.

Using `\:\:class` is safer than relying on a string. It does adapt if the class's name or its namespace is changed'. It is also faster, though it is a micro-optimisation.

It is introduced in PHP 5.5.

```php
<?php

use A\B\C as UsedName;

class foo {
    public function bar( ) {
        echo ClassName::class;
        echo UsedName::class;
    }
}

$f = new Foo( );
$f->bar( );
// displays ClassName
// displays A\B\C

?>
```

Be aware that `\:\:class` is a replacement for __CLASS__ magic constant.

See also Class Constant.

9.8.1 Suggestions

- Use ::class whenever possible. That exclude any dynamic call.

Short name	Php/StaticclassUsage
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*
Php Version	With PHP 5.5 and more recent
Severity	Major
Time To Fix	Slow (1 hour)
Precision	Very high

9.9 @ Operator

@ is the 'no scream' operator : it suppresses error output.

```
<?php

// Set x with incoming value, or else null.
$x = @$_GET['x'];

?>
```

This operator is actually very slow : it will process the error all the way up, and finally decide not to display it. It is often faster to check the conditions first, then run the method without `@`.

You may also set display_error to 0 in the `php.ini` : this will avoid user's error display, but will keep the error in the PHP logs, for later processing.

The only situation where `@` is useful is when a native PHP function displays errors messages when error happens and there is no way to check it from the code.

This is the case with fopen(), stream_socket_server(), token_get_all().

See also Error Control Operators and Five reasons why the shut-op operator should be avoided.

9.9.1 Suggestions

- Remove the @ operator by default

Name	Default	Type	Description
authorizedFunctions	noscream_functions.json	data	Functions that are authorized to sports a @.

Short name	Structures/Noscream
Rulesets	*Analyze*, *Performances*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high
ClearPHP	no-noscream
Examples	*Phinx*, *PhpIPAM*

9.10 Abstract Away

Avoid using PHP native functions that produce data direcly in the code. For example, date() or random_int(). They should be abstracted away in a method, that will be replaced later for testing purposes, or even debugging.

To abstract such calls, place them in a method, and add an interface to this method. Then, create and use those objects.

```
<?php

// abstracted away date
$today = new MyDate();
echo 'Date : '.$today->date('r');

```

(continues on next page)

(continued from previous page)

```
// hard coded date of today : it changes all the time.
echo 'Date : '.date('r');

interface MyCalendar{
    function date($format) : string ;
}

class MyDate implements MyCalendar {
    function date($format) : string { return date('r'); }
}

// Valid implementation, reserved for testing purpose
// This prevents from waiting 4 years for a test.
class MyDateForTest implements MyCalendar {
    function date($format) : string { return date('r', strtotime('2016-02-29 12:00:00
↪')); }
}

?>
```

This analysis targets two API for abstraction : time and random values. Time and date related functions may be replaced by Carbon, Clock, Chronos. Random values may be replaced with RandomLib or a custome interface.

See also Being in control of time in PHP and How to test non-deterministic code.

9.10.1 Suggestions

- Abstract away the calls to native PHP functions, and upgrade the unit tests

Name	Default	Type	Description
abstractableCalls		ini_hash	Functions that shouldn't be called directly, unless in a method.
abstractableClasses		ini_hash	Classes that shouldn't be instantiated directly, unless in a method.

Short name	Patterns/AbstractAway
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.11 Abstract Or Implements

A class must implements all abstract methods of it parent, or be abstract too.

While PHP lints this code, it won't execute it and stop with a Fatal Error : `Class BA contains 1 abstract method and must therefore be declared abstract or implement the remaining methods (A\:\:aFoo)`.

```
<?php

abstract class Foo {
    abstract function FooBar();
}
```

(continues on next page)

(continued from previous page)

```
// This is in another file : php -l would detect it right away

class FooFoo extends Foo {
    // The method is not defined.
    // The class must be abstract, just like Foo
}

?>
```

See also Class Abstraction.

9.11.1 Suggestions

- Implements all the abstract methods of the class
- Make the class abstract

Short name	Classes/AbstractOrImplements
Rulesets	*Analyze*, *LintButWontExec*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Zurmo*

9.12 Abstract Static Methods

Methods cannot be both abstract and static. Static methods belong to a class, and will not be overridden by the child class. For normal methods, PHP will start at the object level, then go up the hierarchy to find the method. With static, it is necessary to mention the name, or use Late Static Binding, with self or static. Hence, it is useless to have an abstract static method : it should be a static method.

A child class is able to declare a method with the same name than a static method in the parent, but those two methods will stay independent.

This is not the case anymore in PHP 7.0+.

```
<?php

abstract class foo {
    // This is not possible
    static abstract function bar() ;
}

?>
```

See also Why does PHP 5.2+ disallow abstract ‘static class methods? <https://stackoverflow.com/questions/999066/why-does-php-5-2-disallow-abstract-static-class-methods>‘_.

9.12.1 Suggestions

- Remove abstract keyword from the method

- Remove static keyword from the method
- Remove the method

Short name	Classes/AbstractStatic
Rulesets	*Analyze*
Php Version	With PHP 7.0 and older
Severity	Minor
Time To Fix	Quick (30 mins)

9.13 Access Protected Structures

It is not allowed to access protected properties or methods from outside the class or its relatives.

```
<?php

class foo {
    protected $bar = 1;
}

$foo = new Foo();
$foo->bar = 2;

?>
```

See also Visibility and Understanding The Concept Of Visibility In Object Oriented PHP.

9.13.1 Suggestions

- Change 'protected' to 'public' to relax the constraint
- Add a getter method to reach the target value
- Remove the access to the protected value and find it another way

Short name	Classes/AccessProtected
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)

9.14 Accessing Private

List of calls to private properties/methods that will compile but yield some fatal error upon execution.

```
<?php

class a {
    private $a;
}

class b extends a {
```

(continues on next page)

(continued from previous page)

```
    function c() {
        $this->a;
    }
}

?>
```

Short name	Classes/AccessPrivate
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)

9.15 Add Default Value

Parameter in methods definition may receive a default value. This allows the called method to set a value when the parameter is omitted.

```
<?php

function foo($i) {
    if (!is_integer($i)) {
        $i = 0;
    }
}

?>
```

See also Function arguments.

9.15.1 Suggestions

- Add a default value for parameters

Short name	Functions/AddDefaultValue
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Zurmo*, *Typo3*

9.16 Adding Zero

Adding 0 is useless, as 0 is the neutral element for addition. Besides, when one of the argument is an integer, PHP triggers a cast to integer.

It is recommended to make the cast explicit with `(int)`.

```
<?php
```

(continues on next page)

(continued from previous page)

```
// Explicit cast
$a = (int) foo();

// Useless addition
$a = foo() + 0;
$a = 0 + foo();

// Also works with minus
$b = 0 - $c; // drop the 0, but keep the minus
$b = $c - 0; // drop the 0 and the minus

$a += 0;
$a -= 0;

?>
```

Adding zero is also reported when the zero is a defined constants.

If it is used to type cast a value to integer, then casting with `(int)` is clearer.

9.16.1 Suggestions

- Remove the +/- 0, may be the whole assignation
- Use an explicit type casting operator (int)

Short name	Structures/AddZero
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
ClearPHP	no-useless-math
Examples	*Thelia*, *OpenEMR*

9.17 Aliases Usage

PHP manual recommends to avoid function aliases.

Some functions have several names, and both may be used the same way. However, one of the names is the main name, and the others are aliases. Aliases may be removed or change or dropped in the future. Even if this is not forecast, it is good practice to use the main name, instead of the aliases.

```
<?php

// official way to count an array
$n = count($array);

// official way to count an array
$n = sizeof($array);

?>
```

Aliases are compiled in PHP, and do not provide any performances over the normal function.

Aliases are more likely to be removed later, but they have been around for a long time.

See documentation : List of function aliases.

9.17.1 Suggestions

- Always use PHP recommended functions

Short name	Functions/AliasesUsage
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
ClearPHP	no-aliases
Examples	*Cleverstyle*, *phpMyAdmin*

9.18 All Uppercase Variables

Usually, global variables are all in uppercase, so as to differentiate them easily. Though, this is not always the case, with examples like $argc, $argv or $http_response_header.

When using custom variables, try to use lowercase `$variables`, `$camelCase`, `$sturdyCase` or `$snake_case`.

```
<?php

// PHP super global, also identified by the initial _
$localVariable = $_POST;

// PHP globals
$localVariable = $GLOBALS['HTTPS'];

?>
```

See also Predefined Variables.

Short name	Variables/VariableUppercase
Rulesets	*Coding Conventions*
Severity	Minor
Time To Fix	Slow (1 hour)

9.19 Already Parents Interface

The same interface is implemented by a class and one of its children.

That way, the child doesn't need to implement the interface, nor define its methods to be an instance of the interface.

```
<?php

interface i {
    function i();
}

```

(continues on next page)

(continued from previous page)

```
class A implements i {
    function i() {
        return __METHOD__;
    }
}

// This implements is useless.
class AB extends A implements i {
    // No definition for function i()
}

// Implements i is understated
class AB extends A {
    // redefinition of the i method
    function i() {
        return __METHOD__.' ';
    }
}

$x = new AB;
var_dump($x instanceof i);
// true

$x = new AC;
var_dump($x instanceof i);
// true

?>
```

9.19.1 Suggestions

- Keep the implements call in the class that do implements the methods. Remove it from the children classes.

Short name	Interfaces/AlreadyParentsInterface
Rulesets	*Analyze*, *Suggestions*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*WordPress*, *Thelia*

9.20 Already Parents Trait

Trait is already used a parent's class or trait. There is no use to include it a second time.

```
<?php

trait ta {
    use tb;
}

trait t1 {
    use ta;
```

(continues on next page)

(continued from previous page)

```
    use tb; // also used by ta
}

class b {
    use t1; // also required by class c
    use ta; // also required by trait t1
}

class c extends b {
    use t1;
}

?>
```

See also Traits.

9.20.1 Suggestions

- Eliminate one of the trait request

Short name	Traits/AlreadyParentsTrait
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.21 Altering Foreach Without Reference

Foreach() loop that should use a reference.

When using a foreach loop that modifies the original source, it is recommended to use referenced variables, rather than access the original value with $source[$index].

Using references is then must faster, and easier to read.

```
<?php

// Using references in foreach
foreach($source as $key => &$value) {
    $value = newValue($value, $key);
}

// Avoid foreach : use array_map
$source = array_walk($source, 'newValue');
    // Here, $key MUST be the second argument or newValue

// Slow version to update the array
foreach($source as $key => &$value) {
    $source[$key] = newValue($value, $key);
}
?>
```

You may also use array_walk() or array_map() (when $key is not used) to avoid the use of foreach.

See also foreach.

9.21.1 Suggestions

- Add the reference on the modified blind variable, and avoid accessing the source array

Short name	Structures/AlteringForeachWithoutReference
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	use-reference-to-alter-in-foreach
Examples	*Contao*, *WordPress*

9.22 Alternative Syntax Consistence

PHP allows for two syntax : the alternative syntax, and the classic syntax.

The classic syntax is almost always used. When used, the alternative syntax is used in templates.

This analysis reports files that are using both syntax at the same time. This is confusing.

```
<?php

// Mixing both syntax is confusing.
foreach($array as $item) :
    if ($item > 1) {
        print $item elements\n;
    } else {
        print $item element\n;
    }
endforeach;

?>
```

Short name	Structures/AlternativeConsistenceByFile
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)

9.23 Always Anchor Regex

Unanchored regex finds the requested pattern, and leaves room for malicious content.

Without ^ and $, the regex searches for any pattern that satisfies the criteria, leaving any unused part of the string available for arbitrary content. It is recommended to use both anchor

```
<?php

$birthday = getSomeDate($_GET);

// Permissive version : $birthday = '1970-01-01<script>xss();</script>';
if (!preg_match('/\d{4}-\d{2}-\d{2}/', $birthday) {
    error('Wrong data format for your birthday!');
```

(continues on next page)

(continued from previous page)

```
}

// Restrictive version : $birthday = '1970-01-01';
if (!preg_match('/^\d{4}-\d{2}-\d{2}$/', $birthday) {
    error('Wrong data format for your birthday!');
}

echo 'Your birthday is on '.$birthday;

?>
```

Note that $ may be a line ending, still leaving room after it for injection.

```
<?php

$birthday = '1970-01-01'.PHP_EOL.'<script>xss();</script>';

?>
```

This analysis reports false positive when the regex is used to search a pattern in a much larger string. Check if this rule doesn't apply, though.

See also CWE-625: Permissive Regular Expression.

9.23.1 Suggestions

- Add an anchor to the beginning and ending of the string

Short name	Security/AnchorRegex
Rulesets	*Security*
Severity	Major
Time To Fix	Instant (5 mins)

9.24 Always Positive Comparison

Some PHP native functions, such as count(), strlen(), or abs() only returns positive or null values.

When comparing them to 0, the following expressions are always true and should be avoided.

```
<?php

$a = [1, 2, 3];

var_dump(count($a) >= 0);
var_dump(count($a) < 0);

?>
```

9.24.1 Suggestions

- Compare count() to non-zero values

- Use empty()

Short name	Structures/NeverNegative
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)
Examples	*Magento*

9.25 Always Use Function With array_key_exists()

array_key_exists() has been granted a special VM opcode, and is much faster. This applies to PHP 7.4 and more recent.

It requires that array_key_exists() is statically resolved, either with an initial \, or a `use function` expression. This doesn't affect the global namespace.

```
<?php

namespace my/name/space;

// do not forget the 'function' keyword, or it will apply to classes.
use function array_key_exists as foo; // the alias is not necessary, and may be␣
↪omitted.

// array_key_exists is aliased to foo :
$c = foo($a, $b);

// This call requires a fallback to global, and will be slow.
$c = array_key_exists($a, $b);

?>
```

This analysis is related to Php/ShouldUseFunction, and is a special case, that only concerns array_key_exists().

See also Add array_key_exists to the list of specialy compiled functions.

9.25.1 Suggestions

- Use the *use* command for arrray_key_exists(), at the beginning of the script
- Use an initial before array_key_exists()
- Remove the namespace

Short name	Performances/Php74ArrayKeyExists
Rulesets	*Performances*
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)

9.26 Ambiguous Array Index

Indexes should not be defined with different types than int or string.

Array indices only accept integers and strings, so any other type of literal is reported. In fact, `null` is turned into an empty string, booleans are turned into an integer, and real numbers are truncated (not rounded).

```
<?php

$x = [ 1  => 1,
      '1' => 2,
      1.0 => 3,
      true => 4];
// $x only contains one element : 1 => 4

// Still wrong, immediate typecast to 1
$x[1.0]   = 5;
$x[true] = 6;

?>
```

They are indeed distinct, but may lead to confusion.

See also array.

9.26.1 Suggestions

- Only use string or integer as key for an array.
- Use transtyping operator (string) and (int) to make sure of the type

Short name	Arrays/AmbiguousKeys
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*PrestaShop*, *Mautic*

9.27 Ambiguous Static

Methods or properties with the same name, are defined static in one class, and not static in another. This is error prone, as it requires a good knowledge of the code to make it static or not.

Try to keep the methods simple and unique. Consider renaming the methods and properties to distinguish them easily. A method and a static method have probably different responsibilities.

```
<?php

class a {
    function mixedStaticMethod() {}
}

class b {
    static function mixedStaticMethod() {}
```

(continues on next page)

(continued from previous page)

```
}

/... a lot more code later .../

$c->mixedStaticMethod();
// or
$c::mixedStaticMethod();

?>
```

Short name	Classes/AmbiguousStatic
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)

9.28 Ambiguous Visibilities

The properties have the same name, but have different visibilities, across different classes.

While it is legit to have a property with the same name in different classes, it may easily lead to confusion. As soon as the context is need to understand if the property is accessible or not, the readability suffers.

It is recommended to handle the same properties in the same way across classes, even when the classes are not related.

```
<?php

class person {
    public $name;
    private $address;
}

class gangster {
    private $name;
    public $nickname;
    private $address;
}

$someone = Human::load(123);
echo 'Hello, '.$someone->name;

?>
```

9.28.1 Suggestions

- Sync visibilities for both properties, in the different classes
- Use different names for properties with different usages

Short name	Classes/AmbiguousVisibilities
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*Typo3*

9.29 Anonymous Classes

Anonymous classes.

```
<?php

// Anonymous class, available since PHP 7.0
$object = new class { function __construct() { echo __METHOD__; } };

?>
```

Short name	Classes/Anonymous
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and more recent
Severity	Major
Time To Fix	Slow (1 hour)

9.30 Argument Should Be Typehinted

When a method expects objects as argument, those arguments should be typehinted. This way, it provides early warning that a wrong object is being sent to the method.

The analyzer will detect situations where a class, or the keywords 'array' or 'callable'.

```
<?php

// What are the possible classes that have a 'foo' method?
function foo($bar) {
    return $bar->foo();
}

?>
```

Closure arguments are omitted.

See also Type declarations.

9.30.1 Suggestions

- Add the typehint to the function arguments

Short name	Functions/ShouldBeTypehinted
Rulesets	*Typechecks*
Severity	Minor
Time To Fix	Slow (1 hour)
ClearPHP	always-typehint
Examples	*Dolphin*, *Mautic*

9.31 Array_Fill() With Objects

array_fill() fills an array with identical objects, not copies nor clones. This means that all the filled objects are a reference to the same object. Changing one of them will change any of them.

Make sure this is the intended effect in the code.

```
<?php

$x = new StdClass();
$array = array_fill(0, 10, $x);

$array[3]->y = Set in object #3;

// displays Set in object #3;
echo $array[5]->y;

?>
```

This applies to array_pad() too. It doesn't apply to array_fill_keys(), as objects will be cast to a string before usage in this case.

9.31.1 Suggestions

- Use a loop to fill in the array with cloned() objects.

Short name	Structures/ArrayFillWithObjects
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.32 Array_merge Needs Array Of Arrays

When collecting data to feed array_merge(), use an array of array as default value. ``array(`array()) <https://www.php.net/array>`_`` is the neutral value for array_merge();

This analysis also reports when the used types are not an array : array_merge() does not accept scalar values, but only arrays.

```
<?php

// safe default value
$a = array(array());
```

(continues on next page)

(continued from previous page)

```
// when $list is empty, it is
foreach($list as $l) {
    $a[] = $l;
}
$b = array_merge($a);

?>
```

Since PHP 7.4, it is possible to call array_merge() without an argument : this means the default value may an empty array. This array shall not contain scalar values.

See also array_merge.

9.32.1 Suggestions

- Use `` `array(array())` `` or `` `[[]]` `` as default value for array_merge()
- Remove any non-array value from the values in the default array

Short name	Structures/ArrayMergeArrayArray
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.33 Assert Function Is Reserved

Avoid defining an `assert` function in namespaces.

While they work fine when the assertions are active (`zend.assertions=1`), calls to unqualified `assert` are optimized away when assertions are not active.

Since PHP 7.3, a fatal error is emitted : ``Defining a custom `assert() <https://www.php.net/assert>`_ function is deprecated, as the function has special semantics``.

```
<?php
//      Run this with zend.assertions=1 and
// Then run this with zend.assertions=0

namespace Test {
    function assert() {
        global $foo;

        $foo = true;
    }
}

namespace Test {
    assert();

    var_dump(isset($foo));
}

?>
```

See also assert and User-defined assert function is optimized away with zend.assertions=-1.

9.33.1 Suggestions

- Rename the custom function with another name

Short name	Php/AssertFunctionIsReserved
Rulesets	*Analyze*, *CompatibilityPHP73*
Severity	Critical
Time To Fix	Slow (1 hour)

9.34 Assign And Compare

Assignation has a lower precedence than comparison. As such, the assignation always happens after the comparison. This leads to the comparison being stored in the variable, and not the value being compared.

```
<?php

if ($id = strpos($string, $needle) !== false) {
    // $id now contains a boolean (true or false), but not the position of the
↪$needle.
}

// probably valid comparison, as $found will end up being a boolean
if ($found = strpos($string, $needle) === false) {
    doSomething();
}

// always valid comparison, with parenthesis
if (($id = strpos($string, $needle)) !== false) {
    // $id now contains a boolean (true or false), but not the position of the
↪$needle.
}

// Being a lone instruction, this is always valid : there is no double usage with if␣
↪condition
$isFound = strpos($string, $needle) !== false;


?>
```

See also Operator Precedence.

9.34.1 Suggestions

- Use parenthesis
- Separate assignation and comparison
- Drop assignation or comparison

Short name	Structures/AssigneAndCompare
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.35 Assign Default To Properties

Properties may be assigned default values at declaration time. Such values may be later modified, if needed.

```
<?php

class foo {
    private $propertyWithDefault = 1;
    private $propertyWithoutDefault;
    private $propertyThatCantHaveDefault;

    public function __construct() {
        // Skip this extra line, and give the default value above
        $this->propertyWithoutDefault = 1;

        // Static expressions are available to set up simple computation at␣
↪definition time.
        $this->propertyWithoutDefault = OtherClass::CONSTANT + 1;

        // Arrays, just like scalars, may be set at definition time
        $this->propertyWithoutDefault = [1,2,3];

        // Objects or resources can't be made default. That is OK.
        $this->propertyThatCantHaveDefault = fopen('/path/to/file.txt');
        $this->propertyThatCantHaveDefault = new Fileinfo();
    }
}

?>
```

Default values will save some instructions in the constructor, and makes the value obvious in the code.

9.35.1 Suggestions

- Add a default value whenever possible. This is easy for scalars, and array()

Short name	Classes/MakeDefault
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Instant (5 mins)
ClearPHP	use-properties-default-values
Examples	*LiveZilla*, *phpMyAdmin*

9.36 Assign With And

The lettered logical operators yield to assignation. It may collect less information than expected.

It is recommended to use the &&, ^ and || operators, instead of and, or and xor, to prevent confusion.

```
<?php

// The expected behavior is
// The following are equivalent
 $a =  $b  && $c;
 $a = ($b && $c);

// The unexpected behavior is
// The following are equivalent
 $a = $b  and $c;
($a = $b) and $c;

?>
```

See also Operator Precedence.

9.36.1 Suggestions

- Always use symbol && rather than letter and
- To be safe, add parenthesis to enforce priorities

Short name	Php/AssignAnd
Rulesets	*Analyze*, *CI-checks*
Severity	Critical
Time To Fix	Quick (30 mins)
Examples	*xataface*

9.37 Assigned Twice

The same variable is assigned twice in the same function.

While this is possible and quite common, it is also a good practice to avoid changing a value from one literal to another. It is far better to assign the new value to

Incremental changes to a variables are not reported here.

```
<?php

function foo() {
    // incremental changes of $a;
    $a = 'a';
    $a++;
    $a = uppercase($a);

    $b = 1;
    $c = bar($b);
    // B changed its purpose. Why not call it $d?
```

(continues on next page)

(continued from previous page)

```
    $b = array(1,2,3);

    // This is some forgotten debug
    $e = $config->getSomeList();
    $e = array('OneElement');
}

?>
```

Short name	Variables/AssignedTwiceOrMore
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.38 Assumptions

Assumptions in the code, that leads to possible bugs.

Some conditions may be very weak, and lead to errors. For example, the code below checks that the variable *$a* is not null, then uses it as an array. There is no relationship between 'not null' and 'being an array', so this is an assumption.

```
<?php

// Assumption : if $a is not null, then it is an array. This is not always the case.
function foo($a) {
    if ($a !== null) {
        echo $a['name'];
    }
}

// Assumption : if $a is not null, then it is an array. Here, the typehint will␣
↪ensure that it is the case.
// Although, a more readable test is is_array()
function foo(?array $a) {
    if ($a !== null) {
        echo $a['name'];
    }
}

?>
```

See also From assumptions to assertions.

9.38.1 Suggestions

-

Short name	Php/Assumptions
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.39 Autoappend

Appending a variable to itself leads to enormous usage of memory.

```php
<?php

// Always append a value to a distinct variable
foreach($a as $b) {
    $c[] = $b;
}

// This copies the array to itself, and double the size each loop
foreach($a as $b) {
    $c[] = $c;
}
?>
```

9.39.1 Suggestions

- Change the variable in the append, on the left
- Change the variable in the append, on the right

Short name	Performances/Autoappend
Rulesets	*Performances*
Severity	Minor
Time To Fix	Quick (30 mins)

9.40 Avoid Concat In Loop

Concatenations inside a loop generate a lot of temporary variables. They are accumulated and tend to raise the memory usage, leading to slower performances.

It is recommended to store the values in an array, and then use implode() on that array to make the concatenation at once. The effect is positive when the source array has at least 50 elements.

```php
<?php

// Concatenation in one operation
$tmp = array();
foreach(data_source() as $data) {
    $tmp[] = $data;
}
$final = implode('', $tmp);

// Concatenation in many operations
foreach(data_source() as $data) {
    $final .= $data;
}

?>
```

The same doesn't apply to addition and multiplication, with array_sum() and array_multiply(), as those operations work on the current memory allocation, and don't need to allocate new memory at each step.

See also PHP 7 performance improvements (3/5): Encapsed strings optimization.

9.40.1 Suggestions

- Collect all pieces in an array, then implode() the array in one call.

Short name	Performances/NoConcatInLoop
Rulesets	*Performances*, *Top10*
Severity	Major
Time To Fix	Slow (1 hour)
Examples	*SuiteCrm*, *ThinkPHP*

9.41 Avoid Large Array Assignation

Avoid setting large arrays to local variables. This is done every time the function is called.

There are different ways to avoid this : inject the array, build the array once. Using an constant or even a global variable is faster.

The effect on small arrays (less than 10 elements) is not significant. Arrays with 10 elements or more are reported here. The effect is also more important on functions that are called often, or within loops.

```
<?php

// with constants, for functions
const ARRAY = array(1,2,3,4,5,6,7,8,9,10,11);
function foo() {
    $array = ARRAY;
    //more code
}

// with class constants, for methods
class x {
    const ARRAY = array(1,2,3,4,5,6,7,8,9,10,11);
    function foo() {
        $array = self::ARRAY;
        //more code
    }
}

// with properties, for methods
class x {
    private $array = array(1,2,3,4,5,6,7,8,9,10,11);

    function foo() {
        $array = $this->array;
        //more code
    }
}

// injection, leveraging default values
```

(continues on next page)

(continued from previous page)

```
function foo($array = array(1,2,3,4,5,6,7,8,9,10,11)) {
    //more code
}

// local cache with static
function foo() {
    static $array;
    if ($array === null) {
        $array = array(1,2,3,4,5,6,7,8,9,10,11);
    }

    //more code
}

// Avoid creating the same array all the time in a function
class x {
    function foo() {
        // assign to non local variable is OK.
        // Here, to a property, though it may be better in a __construct or as␣
↪default values
        $this->s = array(1,2,3,4,5,6,7,8,9,10,11);

        // This is wasting resources, as it is done each time.
        $array = array(1,2,3,4,5,6,7,8,9,10,11);
    }
}

?>
```

Short name	Structures/NoAssignationInFunction
Rulesets	*Performances*
Severity	Minor
Time To Fix	Slow (1 hour)

9.42 Avoid Optional Properties

Avoid optional properties, to prevent littering the code with existence checks.

When a property has to be checked once for existence, it is safer to check it each time. This leads to a decrease in readability and a lot of checks added to the code.

Either make sure the property is set with an actual object rather than with null, or use a null object. A null object offers the same interface than the expected object, but does nothing. It allows calling its methods, without running into a Fatal error, nor testing it.

```
<?php

// Example is courtesy 'The Coding Machine' : it has been adapted from its original␣
↪form. See link below.

class MyMailer {
    private $logger;
```

(continues on next page)

(continued from previous page)

```
    public function __construct(LoggerInterface $logger = null) {
        $this->logger = $logger;
    }

    private function sendMail(Mail $mail) {
        // Since $this->logger may be null, it must be tested anytime it is used.
        if ($this->logger) {
            $this->logger->info('Mail successfully sent.');
        }
    }
}

?>
```

See also Avoid optional services as much as possible, The Null Object Pattern – Polymorphism in Domain Models, and Practical PHP Refactoring: Introduce Null Object.

9.42.1 Suggestions

- Use a null object to fill any missing value
- Make sure the property is set at constructor time

Short name	Classes/AvoidOptionalProperties
Rulesets	*Analyze*
Severity	Major
Time To Fix	Slow (1 hour)
Examples	*ChurchCRM*, *Dolibarr*

9.43 Avoid Parenthesis

Avoid Parenthesis for language construct. Languages constructs are a few PHP native elements, that looks like functions but are not.

Among other distinction, those elements cannot be directly used as variable function call, and they may be used with or without parenthesis.

```
<?php

// normal usage of include
include 'file.php';

// This looks like a function and is not
include('file2.php');

?>
```

The usage of parenthesis actually give some feeling of comfort, it won't prevent PHP from combining those argument with any later operators, leading to unexpected results.

Even if most of the time, usage of parenthesis is legit, it is recommended to avoid them.

Short name	Structures/PrintWithoutParenthesis
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.44 Avoid Real

PHP has two float data type : real and double. `real` is rarely used, and might be deprecated in PHP 7.4.

To prepare code, avoid using is_real() and the `(real)` typecast.

```
<?php

// safe way to check for float
if (!is_float($a)) {
    $a = (float) $a;
}

// Avoid doing that
if (!is_real($a)) {
    $a = (real) $a;
}

?>
```

See also PHP RFC: Deprecations for PHP 7.4.

9.44.1 Suggestions

- Replace is_real() by is_float()
- Replace (real) by (float)

Short name	Php/AvoidReal
Rulesets	*Suggestions*, *Top10*
Severity	Minor
Time To Fix	Quick (30 mins)

9.45 Avoid Self In Interface

Self and Parent are tricky when used in an interface.

`self` refers to the current interface or its extended parents : as long as the constant is defined in the interface family, this is valid. On the other hand, when `self` refers to the current class, the resolution of names will happen at execution time, leading to confusing results.

`parent` has the same behavior than `self`, except that it doesn't accept to be used inside an interface, as it will yield an error. This is one of those error that lint but won't execute in certain conditions.

`Static` can't be used in an interface, as it needs to be resolved at call time anyway.

```
<?php

interface i extends ii {
    // This 'self' is valid : it refers to the interface i
    public const I = self::I2 + 2;

    // This 'self' is also valid, as it refers to interface ii, which is a part of␣
↪interface i
    public const I2 = self::IP + 4;

    // This makes interface i dependant on the host class
    public const I3 = parent::A;
}

?>
```

See also Scope Resolution Operator (::).

9.45.1 Suggestions

- Use a fully qualified namespace instead of self
- Use a locally defined constant, so self is a valid reference

Short name	Interfaces/AvoidSelfInInterface
Rulesets	*ClassReview*
Severity	Critical
Time To Fix	Slow (1 hour)

9.46 Avoid Substr() One

Use array notation `$string[$position]` to reach a single byte in a string.

There are two ways to access a byte in a string : substr() and `$v[$pos]`.

The second style is more readable. It may be up to four times faster, though it is a micro-optimization. It is recommended to use it.

PHP 7.1 also introduces the support of negative offsets as string index : negative offset are also reported.

```
<?php

$string = 'abcde';

echo substr($string, $pos, 1);
echo $string[$pos];

echo mb_substr($string, $pos, 1);

// when $pos = 1
// displays bbb
// when $pos = 2
// displays ??
```

(continues on next page)

(continued from previous page)

```

?>
```

Beware that substr() and `$v[$pos]` are similar, while mb_substr() is not. The first function works on bytes, while the latter works on characters.

9.46.1 Suggestions

- Replace substr() with the array notations for strings.
- Replace substr() with a call to mb_substr().

Short name	Structures/NoSubstrOne
Rulesets	*Analyze*, *Performances*, *CompatibilityPHP71*, *Suggestions*, *Top10*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*ChurchCRM*, *LiveZilla*

9.47 Avoid Those Hash Functions

The following cryptography algorithms are considered insecure, and should be replaced with new and more performant algorithms.

`MD2`, `MD4`, `MD5`, `SHA0`, `SHA1`, `CRC`, `DES`, `3DES`, `RC2`, `RC4`.

When possible, avoid using them, may it be as PHP functions, or hashing function configurations (mcrypt, hash…).

```
<?php

// Weak cryptographic algorithm
echo md5('The quick brown fox jumped over the lazy dog.');

// Weak crypotgraphic algorthim, used with a modern PHP extension (easier to update)
echo hash('md5', 'The quick brown fox jumped over the lazy dog.');

// Strong crypotgraphic algorthim, used with a modern PHP extension
echo hash('sha156', 'The quick brown fox jumped over the lazy dog.');

?>
```

Weak cryptography is commonly used for hashing values when caching them. In such cases, security is not a primary concern. However, it may later become such, when hackers get access to the cache folders, or if the cached identifier is published. As a preventive protection, it is recommended to always use a secure hashing function.

See also Secure Hash Algorithms.

9.47.1 Suggestions

- Keep the current crypto, and add a call to a stronger one.
- Change the crypto for a more modern one and update the related databases

Short name	Security/AvoidThoseCrypto
Rulesets	*Security*
Severity	Major
Time To Fix	Quick (30 mins)

9.48 Avoid Using stdClass

`stdClass` is the default class for PHP. It is instantiated when PHP needs to return a object, but no class is specifically available.

It is recommended to avoid instantiating this class, nor use it is any way.

```
<?php

$json = '{a:1,b:2,c:3}';
$object = json_decode($json);
// $object is a stdClass, as returned by json_decode

// Fast building of $o
$a = [];
$a['a'] = 1;
$a['b'] = 2;
$a['c'] = 3;
json_encode( (object) $a);

// Slow building of $o
$o = new stdClass();
$o->a = 1;
$o->b = 2;
$o->c = 3;
json_encode($o);

?>
```

If you need a `stdClass` object, it is faster to build it as an array, then cast it, than instantiate `stdClass`. This is a micro-optimisation.

9.48.1 Suggestions

- Create a custom class to handle the properties

Short name	Php/UseStdclass
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)

9.49 Avoid array_push()

array_push() is slower than the [] operator.

This is also true if the [] operator is called several times, while array_push() may be called only once. And using count after the push is also faster than collecting array_push() return value.

```
<?php

$a = [1,2,3];
// Fast version
$a[] = 4;

$a[] = 5;
$a[] = 6;
$a[] = 7;
$count = count($a);

// Slow version
array_push($a, 4);
$count = array_push($a, 5,6,7);

// Multiple version :
$a[] = 1;
$a[] = 2;
$a[] = 3;
array_push($a, 1, 2, 3);


?>
```

This is a micro-optimisation.

9.49.1 Suggestions

- Use the [] operator

Short name	Performances/AvoidArrayPush
Rulesets	*Performances*
Severity	Minor
Time To Fix	Instant (5 mins)

9.50 Avoid array_unique()

The native function array_unique() is much slower than using other alternatives, such as array_count_values(), array_flip()/array_keys(), or even a foreach() loops.

```
<?php

// using array_unique()
$uniques = array_unique($someValues);

// When values are strings or integers
$uniques = array_keys(array_count_values($someValues));
$uniques = array_flip(array_flip($someValues))

//even some loops are faster.
```

(continues on next page)

(continued from previous page)

```
$uniques = [];
foreach($someValues as $s) {
    if (!in_array($uniques, $s)) {
        $uniques[] $s;
    }
}

?>
```

See also array_unique.

9.50.1 Suggestions

- Upgrade to PHP 7.2
- Use an alternative way to make values unique in an array, using array_count_values(), for example.

Short name	Structures/NoArrayUnique
Rulesets	*Performances*
Php Version	7.2-
Severity	Minor
Time To Fix	Quick (30 mins)

9.51 Avoid get_class()

`get_class()` should be replaced with the `instanceof` operator to check the class of an object.

`get_class()` only compares the full namespace name of the object's class, while `instanceof` actually resolves the name, using the local namespace and aliases.

```
<?php

    use Stdclass as baseClass;

    function foo($arg) {
        // Slow and prone to namespace errors
        if (get_class($arg) === 'Stdclass') {
            // doSomething()
        }
    }

    function bar($arg) {
        // Faster, and uses aliases.
        if ($arg instanceof baseClass) {
            // doSomething()
        }
    }
?>
```

See also get_class and Instanceof.

Short name	Structures/UseInstanceof
Rulesets	*Analyze*, *Analyze*, *CI-checks*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.52 Avoid glob() Usage

glob() and scandir() sorts results by default. When that kind of sorting is not needed, save some time by requesting NOSORT with those functions.

Besides, whenever possible, use scandir() instead of glob().

```
<?php

// Scandir without sorting is the fastest.
scandir('docs/', SCANDIR_SORT_NONE);

// Scandir sorts files by default. Same as above, but with sorting
scandir('docs/');

// glob sorts files by default. Same as below, but no sorting
glob('docs/*', GLOB_NOSORT);

// glob sorts files by default. This is the slowest version
glob('docs/*');

?>
```

Using opendir() and a while loop may be even faster.

This analysis skips scandir() and glob() if they are expliciely configured with flags (aka, sorting is explicitly needed).

glob() accepts wildchar, such as `*`, that may not easily replaced with scandir() or opendir().

See also Putting glob to the test, How to list files recursively in a directory with PHP iterators and glob://.

9.52.1 Suggestions

- Use FilesystemIterator, DirectoryIterator classes.
- Use `RegexIterator` to filter any unwanted results from `FilesystemIterator`.
- Use `glob` protocol for files : $it = new DirectoryIterator('glob://path/to/examples/*.php');

Short name	Performances/NoGlob
Rulesets	*Performances*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Phinx*, *NextCloud*

9.53 Avoid mb_dectect_encoding()

mb_dectect_encoding() is bad at guessing encoding.

For example, UTF-8 and ISO-8859-1 share some common characters : when a string is build with them it is impossible to differentiate the actual encoding.

```
<?php

$encoding = mb_encoding_detect($_GET['name']);

?>
```

See also mb_encoding_detect, PHP vs. The Developer: Encoding Character Sets, DPC2019: Of representation and interpretation: A unified theory - Arnout Boks.

9.53.1 Suggestions

- Store and transmit the data format

Short name	Php/AvoidMbDectectEncoding
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.54 Avoid option arrays in constructors

Avoid option arrays in constructors. Use one parameter per injected element.

```
<?php

class Foo {
    // Distinct arguments, all typehinted if possible
    function __constructor(A $a, B $b, C $c, D $d) {
        $this->a = $a;
        $this->b = $b;
        $this->c = $c;
        $this->d = $d;
    }
}

class Bar {
    // One argument, spread over several properties
    function __constructor(array $options) {
        $this->a = $options['a'];
        $this->b = $options['b'];
        $this->c = $options['c'];
        $this->d = $options['d'];
    }
}

?>
```

See also Avoid option arrays in constructors.

9.54.1 Suggestions

- Spread the options in the argument list, one argument each
- Use a configuration class, that hold all the elements with clear names, instead of an array

Short name	Classes/AvoidOptionArrays
Rulesets	*Analyze*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.55 Avoid set_error_handler $context Argument

Avoid configuring set_error_handler() with a method that accepts 5 arguments. The last argument, `$errcontext`, is deprecated since PHP 7.2, and will be removed later.

```
<?php

// setting error_handler with an incorrect closure
set_error_handler(function($errno, $errstr, $errfile, $errline) {});

// setting error_handler with an incorrect closure
set_error_handler(function($errno, $errstr, $errfile, $errline, $errcontext) {});

?>
```

See also set_error_handler();

9.55.1 Suggestions

- Remove the 6th argument of registered handlers.

Short name	Php/AvoidSetErrorHandlerContextArg
Rulesets	*CompatibilityPHP72*
Severity	Major
Time To Fix	Slow (1 hour)
Examples	*shopware*, *Vanilla*

9.56 Avoid sleep()/usleep()

sleep() and usleep() help saturate the web server.

Pausing the script for a specific amount of time means that the Web server is also making all related resources sleep, such as database, sockets, session, etc. This may used to set up a DOS on the server.

```
<?php

$begin = microtime(true);
checkLogin($user, $password);
$end   = microtime(true);
```

(continues on next page)

(continued from previous page)

```

// Making all login checks looks the same
usleep(1000000 - ($end - $begin) * 1000000);

// Any hit on this page now uses 1 second, no matter if load is high or not
// Is it now possible to saturate the webserver in 1 s ?

?>
```

As much as possible, avoid delaying the end of the script.

sleep() and usleep() have less impact in commandline (CLI).

9.56.1 Suggestions

- Add a deadline of usage in the session, and wait past this deadline to start serving again. Until then, abort immediately.
- Use element in the GUI to delay or slow usage.

Short name	Security/NoSleep
Rulesets	*Security*
Severity	Minor
Time To Fix	Quick (30 mins)

9.57 Bad Constants Names

PHP's manual recommends that developer do not use constants with the convention __NAME__. Those are reserved for PHP future use.

For example, __TRAIT__ recently appeared in PHP, as a magic constant. In the future, other may appear.

```
<?php

const __MY_APP_CONST__ = 1;

const __MY_APP_CONST__ = 1;

define('__MY_OTHER_APP_CONST__', 2);

?>
```

The analyzer will report any constant which name is __.*.__, or even _.*_ (only one underscore).

See also Constants.

9.57.1 Suggestions

- Avoid using names that doesn't comply with PHP's convention

Short name	Constants/BadConstantnames
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*PrestaShop*, *Zencart*

9.58 Bad Typehint Relay

A bad typehint relay happens where a type hinted argument is relayed to a parameter with another typehint. This will lead to a Fatal error, and stop the code. This is possibly a piece of dead code.

```
<?php

// the $i argument is relayed to bar, which is expecting a string.
function foo(int $i) : string {
    return bar($i);
}

// the return value for the bar function is not compatible with the one from foo;
function bar(string $s) : int {
    return (int) $string + 1;
}

?>
```

It is recommended to harmonize the typehint, so the two functions are still compatible.

9.58.1 Suggestions

- Harmonize the typehint so they match one with the other.
- Remove dead code
- Apply type casting before calling the next function, or return value

Short name	Functions/BadTypehintRelay
Rulesets	*Typechecks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.59 Bail Out Early

When using conditions, it is recommended to quit in the current context, and avoid else clause altogether.

The main benefit is to make clear the method applies a condition, and stop immediately when it is not satisfied. The main sequence is then focused on the actual code.

This works with the `break`, `continue`, `throw` and `goto` keywords too, depending on situations.

```php
<?php

// Bailing out early, low level of indentation
function foo1($a) {
    if ($a > 0) {
        return false;
    }

    $a++;
    return $a;
}

// Works with continue too
foreach($array as $a => $b) {
    if ($a > 0) {
        continue false;
    }

    $a++;
    return $a;
}

// No need for else
function foo2($a) {
    if ($a > 0) {
        return false;
    } else {
        $a++;
    }

    return $a;
}

// No need for else : return goes into then.
function foo3($a) {
    if ($a < 0) {
        $a++;
    } else {
        return false;
    }

    return $a;
}

// Make a return early, and make the condition visible.
function foo3($a) {
    if ($a < 0) {
        $a++;
        methodcall();
        functioncall();
    }
}

?>
```

See also Avoid nesting too deeply and return early (part 1) and Avoid nesting too deeply and return early (part 2).

9.59.1 Suggestions

- Detect errors, and then, return as soon as possible.
- When a if…then branches are unbalanced, test for the small branch, finish it with return. Then keep the other branch as the main code.

Short name	Structures/BailOutEarly
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*OpenEMR*, *opencfp*

9.60 Binary Glossary

List of all the integer values using the binary format.

```
<?php

$a = 0b10;
$b = 0B0101;

?>
```

Short name	Type/Binary
Rulesets	*CompatibilityPHP53*
Php Version	With PHP 5.4 and more recent
Severity	Major
Time To Fix	Quick (30 mins)

9.61 Bracketless Blocks

PHP allows one liners as for(), foreach(), while(), do/while() loops, or as then/else expressions.

It is generally considered a bad practice, as readability is lower and there are non-negligible risk of excluding from the loop the next instruction.

```
<?php

// Legit one liner
foreach(range('a', 'z') as $letter) ++$letterCount;

// More readable version, even for a one liner.
foreach(range('a', 'z') as $letter) {
    ++$letterCount;
}

?>
```

switch() cannot be without bracket.

Short name	Structures/Bracketless
Rulesets	*Coding Conventions*
Severity	Minor
Time To Fix	Instant (5 mins)

9.62 Break Outside Loop

Starting with PHP 7, break or continue that are outside a loop (for, foreach(), do… ‘while() <https://www.php.net/manual/en/control-structures.while.php>‘_, while()) or a switch() statement won’t compile anymore.

It is not possible anymore to include a piece of code inside a loop that will then break.

```
<?php

    // outside a loop : This won't compile
    break 1;

    foreach($array as $a) {
        break 1; // Compile OK

        break 2; // This won't compile, as this break is in one loop, and not 2
    }

    foreach($array as $a) {
        foreach($array2 as $a2) {
            break 2; // OK in PHP 5 and 7
        }
    }
?>
```

Short name	Structures/BreakOutsideLoop
Rulesets	*Analyze*, *CompatibilityPHP70*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.63 Break With 0

Cannot break 0, as this makes no sense. Break 1 is the minimum, and is the default value.

```
<?php
    // Can't break 0. Must be 1 or more, depending on the level of nesting.
    for($i = 0; $i < 10; $i++) {
        break 0;
    }

    for($i = 0; $i < 10; $i++) {
        for($j = 0; $j < 10; $j++) {
            break 2;
        }
    }
```

(continues on next page)

(continued from previous page)

```

?>
```

Short name	Structures/Break0
Rulesets	*CompatibilityPHP53*
Php Version	With PHP 5.4 and older
Severity	Minor
Time To Fix	Quick (30 mins)

9.64 Break With Non Integer

When using a break, the argument of the operator must be a positive non-null integer literal or be omitted.

Other values were acceptable in PHP 5.3 and previous version, but this is now reported as an error.

```
<?php
    // Can't break $a, even if it contains an integer.
    $a = 1;
    for($i = 0; $i < 10; $i++) {
        break $a;
    }

    // can't break on float
    for($i = 0; $i < 10; $i++) {
        for($j = 0; $j < 10; $j++) {
            break 2.2;
        }
    }

?>
```

Short name	Structures/BreakNonInteger
Rulesets	*CompatibilityPHP54*
Php Version	With PHP 5.4 and older
Severity	Minor
Time To Fix	Quick (30 mins)

9.65 Buried Assignation

Those assignations are buried in the code, and placed in unexpected situations.

They are difficult to spot, and may be confusing. It is advised to place them in a more visible place.

```
<?php

// $b may be assigned before processing $a
$a = $c && ($b = 2);

// Display property p immeiately, but also, keeps the object for later
echo ($o = new x)->p;
```

(continues on next page)

(continued from previous page)

```
// legit syntax, but the double assignation is not obvious.
for($i = 2, $j = 3; $j < 10; $j++) {

}
?>
```

9.65.1 Suggestions

- Extract the assignation and set it on its own line, prior to the current expression.
- Check if the local variable is necessary

Short name	Structures/BuriedAssignation
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*XOOPS*, *Mautic*

9.66 Cache Variable Outside Loop

Avoid recalculating constant values inside the loop.

Do the calculation once, outside the loops, and then reuse the value each time.

One of the classic example if doing `count($array)` in a `for` loop : since the source is constant during the loop, the result of count() is always the same.

```
<?php

$path = '/some/path';
$fullpath = realpath("$path/more/dirs/");
foreach($files as $file) {
    // Only moving parts are used in the loop
    copy($file, $fullpath.$file);
}

$path = '/some/path';
foreach($files as $file) {
    // $fullpath is calculated each loop
    $fullpath = realpath("$path/more/dirs/");
    copy($file, $fullpath.$file);
}

?>
```

Depending on the load of the called method, this may increase the speed of the loop from little to enormously.

9.66.1 Suggestions

- Avoid using blind variables outside loops.

- Store blind variables in local variables or properties for later reuse.

Short name	Performances/CacheVariableOutsideLoop
Rulesets	*Performances*

9.67 Callback Needs Return

When used with array_map() functions, the callback must return something. This return may be in the form of a `return` statement, a global variable or a parameter with a reference. All those solutions extract information from the callback.

```
<?php

// This filters each element
$filtered = array_filter($array, function ($x) {return $x == 2; });

// This return void for every element
$filtered = array_filter($array, function ($x) {return ; });

// costly array_sum()
$sum = 0;
$filtered = array_filter($array, function ($x) use (&$sum) {$sum += $x; });

// costly array_sum()
global $sum = 0;
$filtered = array_filter($array, function () {global $sum; $sum += $x; });

// register_shutown_function() doesn't require any return
register_shutown_function(my_shutdown);

?>
```

The following functions are omitted, as they don't require the return :

- forward_static_call_array()
- forward_static_call()
- register_shutdown_function()
- register_tick_function()

See also array_map.

9.67.1 Suggestions

- Add an explicit return to the callback
- Use *null* to unset elements in an array without destroying the index

Short name	Functions/CallbackNeedsReturn
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)
Examples	*Contao*, *Phpdocumentor*

9.68 Calltime Pass By Reference

PHP doesn't allow when a value is turned into a reference at functioncall, since PHP 5.4.

Either the function use a reference in its signature, either the reference won't pass.

```
<?php

function foo($name) {
    $arg = ucfirst(strtolower($name));
    echo 'Hello '.$arg;
}

$a = 'name';
foo(&$a);

?>
```

9.68.1 Suggestions

- Make the signature of the called method accept references
- Remove the reference from the method call
- Use an object instead of a scalar

Short name	Structures/CalltimePassByReference
Rulesets	*CompatibilityPHP54*
Php Version	With PHP 5.4 and older
Severity	Minor
Time To Fix	Quick (30 mins)

9.69 Can't Count Non-Countable

Count() emits an error when it tries to count scalars or objects what don't implement Countable interface.

```
<?php

// Normal usage
$a = array(1,2,3,4);
echo count($a).items\n;

// Error emiting usage
$a = '1234';
echo count($a).chars\n;

// Error emiting usage
echo count($unsetVar).elements\n;

?>
```

See also Warn when counting non-countable types.

Short name	Structures/CanCountNonCountable
Rulesets	*CompatibilityPHP72*
Severity	Major
Time To Fix	Quick (30 mins)

9.70 Can't Extend Final

It is not possible to extend final classes.

Since PHP fails with a fatal error, this means that the extending class is probably not used in the rest of the code. Check for dead code.

```
<?php
    // File Foo
    final class foo {
        public final function bar() {
            // doSomething
        }
    }
?>
```

In a separate file :

```
<?php
    // File Bar
    class bar extends foo {

    }
?>
```

See also Final Keyword.

9.70.1 Suggestions

- Remove the final keyword
- Remove the extending class

Short name	Classes/CantExtendFinal
Rulesets	*Analyze*, *Dead code*
Severity	Critical
Time To Fix	Instant (5 mins)

9.71 Can't Throw Throwable

Classes extending `Throwable` can't be thrown. The same applies to interfaces.

Although this code lints, PHP throws a Fatal error when executing or including it : `` Class fooThrowable cannot implement interface `Throwable <https://www.php.net/manual/en/class.throwable.php>`_, extend Exception or Error instead. ``

```
<?php

// This is the way to go
class fooException extends \Exception { }

// This is not possible and a lot of work
class fooThrowable implements \throwable { }

?>
```

See also Throwable, Exception and Error.

9.71.1 Suggestions

- Extends the Exception class
- Extends the Error class

Short name	Exceptions/CantThrow
Rulesets	*Analyze*, *LintButWontExec*
Severity	Minor
Time To Fix	Slow (1 hour)

9.72 Cancel Common Method

A parent method's is too little used in children.

The parent class has a method, which is customised in children classes, though most of the time, those are empty : hence, cancelled.

```
<?php

class x {
    abstract function foo();
    abstract function bar();
}

class y1 extends x {
    function foo() { doSomething(); }
    function bar() { doSomething(); };
}

class y2 extends x {
    // foo is cancelled : it must be written, but has no use.
    function foo() {   }
    function bar() { doSomething(); };
}

?>
```

A threshold of `cancelThreshold` % of the children methods have to be cancelled to report the parent class. By default, it is 75 (or 3 out of 4).

9.72.1 Suggestions

- Drop the common method, and the cancelled methods in the children
- Fill the children's methods with actual code

Name	Default	Type	Description
cancelThreshold	75	integer	Minimal number of cancelled methods to suggest the cancellation of the parent.

Short name	Classes/CancelCommonMethod
Rulesets	*Suggestions*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.73 Cant Implement Traversable

It is not possible to implement the Traversable``interface. The alternative is to implement ``Iterator or IteratorAggregate.

Traversable may be useful when used with instanceof.

```
<?php

// This lints, but doesn't run
class x implements Traversable {

}

if( $argument instanceof Traversable ) {
    // doSomething
}

?>
```

See also Traversable, Iterator and IteratorAggregate..

9.73.1 Suggestions

- Implement Iterator or IteratorAggregate

Short name	Interfaces/CantImplementTraversable
Rulesets	*Analyze*, *LintButWontExec*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.74 Cant Inherit Abstract Method

Inheriting abstract methods was made available in PHP 7.2. In previous versions, it emitted a fatal error.

```
<?php

abstract class A           { abstract function bar(stdClass $x);  }
abstract class B extends A { abstract function bar($x): stdClass; }

//   Fatal error: Can't inherit abstract function A::bar()
?>
```

See also PHP RFC: Allow abstract function override.

9.74.1 Suggestions

- Avoid inheriting abstract methods for compatibility beyond 7.2 (and older)

Short name	Classes/CantInheritAbstractMethod
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.2 and more recent
Severity	Critical
Time To Fix	Quick (30 mins)

9.75 Cant Instantiate Class

When constructor is not public, it is not possible to instantiate such a class. Either this is a conception choice, or there are factories to handle that. Either way, it is not possible to call new on such class.

PHP reports an error similar to this one : 'Call to private Y::__construct() from invalid context'.

```
<?php

//This is the way to go
$x = X::factory();

//This is not possible
$x = new X();

class X {
    //This is also the case with proctected __construct
    private function __construct() {}

    static public function factory() {
        return new X();
    }
}

?>
```

See also In a PHP5 class, when does a private constructor get called?, Named Constructors in PHP and PHP Constructor Best Practices And The Prototype Pattern.

Short name	Classes/CantInstantiateClass
Rulesets	*Analyze*
Severity	Critical
Time To Fix	Quick (30 mins)
Examples	*WordPress*

9.76 Cant Use Return Value In Write Context

empty() used to work only on data containers, such as variables. Until PHP 5.5, it was not possible to use directly expressions, such as functioncalls, inside an empty() function call : they were met with a 'Can't use function return value in write context' fatal error.

```
<?php

function foo($boolean) {
    return $boolean;
}

// Valid since PHP 5.5
echo empty(foo(true)) : 'true' : 'false';

?>
```

This also applies to methodcalls, static or not.

See also Cant Use Return Value In Write Context.

Short name	Php/CantUseReturnValueInWriteContext
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*
Php Version	With PHP 5.5 and more recent
Severity	Major
Time To Fix	Quick (30 mins)

9.77 Case Insensitive Constants

PHP constants may be case insensitive, when defined with define() and the third argument.

This feature is deprecated since PHP 7.3 and will be removed in PHP 8.0.

```
<?php

// case sensitive
define('A', 1);

// case insensitive
define('B', 1, true);

echo A;
// This is not possible
//echo a;
```

(continues on next page)

(continued from previous page)

```
// both possible
echo B;
echo b;

?>
```

See also define.

Short name	Constants/CaseInsensitiveConstants
Rulesets	*CompatibilityPHP73*
Severity	Critical
Time To Fix	Slow (1 hour)

9.78 Cast To Boolean

This expression may be reduced by casting to boolean type.

```
<?php

$variable = $condition == 'met' ? 1 : 0;
// Same as
$variable = (bool) $condition == 'met';

$variable = $condition == 'met' ? 0 : 1;
// Same as (Note the condition inversion)
$variable = (bool) $condition != 'met';
// also, with an indentical condition
$variable = !(bool) $condition == 'met';

// This also works with straight booleans expressions
$variable = $condition == 'met' ? true : false;
// Same as
$variable = $condition == 'met';

?>
```

9.78.1 Suggestions

- Remove the old expression and use `(bool)` operator instead
- Change the target values from true/false, or 0/1 to non-binary values, like strings or integers beyond 0 and 1.
- Complete the current branches with other commands

Short name	Structures/CastToBoolean
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*MediaWiki*, *Dolibarr*

9.79 Cast Unset Usage

Usage of the *(unset)* cast operator. It is removed in PHP 8.0, and was deprecated since PHP 7.2.0.

```
<?php

$a = 1;
(unset) $a;

// functioncall is OK
unset($a);

?>
```

See also Unset casting.

9.79.1 Suggestions

- Replace *(unset)* with a call to unset().
- Remove the unset call altogether.

Short name	Php/CastUnsetUsage
Rulesets	*CompatibilityPHP80*
Php Version	8.0-
Severity	Minor
Time To Fix	Quick (30 mins)

9.80 Casting Ternary

Type casting has a precedence over ternary operator, and is applied first. When this happens, the condition is cast, although it is often useless as PHP will do it if needed.

This applies to the ternary operator, the coalesce operator ?: and the null-coalesce operator ??.

```
<?php
    $a = (string) $b ? 3 : 4;
    $a = (string) $b ?: 4;
    $a = (string) $b ?? 4;
?>
```

The last example generates first an error *Undefined variable: b*, since $b is first cast to a string. The result is then an empty string, which leads to an empty string to be stored into $a. Multiple errors cascade.

See also Operators Precedence.

9.80.1 Suggestions

- Add parenthesis around the ternary operator
- Skip the casting
- Cast in another expression

Short name	Structures/CastingTernary
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)

9.81 Catch Overwrite Variable

The try/catch structure uses some variables that are also in use in this scope. In case of a caught exception, the exception will be put in the catch variable, and overwrite the current value, loosing some data.

```
<?php

// variables and caught exceptions are distinct
$argument = 1;
try {
    methodThatMayRaiseException($argument);
} (Exception $e) {
    // here, $e has been changed to an exception.
}

// variables and caught exceptions are overlapping
$e = 1;
try {
    methodThatMayRaiseException();
} (Exception $e) {
    // here, $e has been changed to an exception.
}

?>
```

It is recommended to use another name for these catch variables.

9.81.1 Suggestions

- Use a standard : only use $e (or else) to catch exceptions. Avoid using them for anything else, parameter, property or local variable.
- Change the variable, and keep the caught exception

Short name	Structures/CatchShadowsVariable
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Instant (5 mins)
ClearPHP	no-catch-overwrite
Examples	*PhpIPAM*, *SuiteCrm*

9.82 Catch Undefined Variable

Always initialize variable before the try block, when they are used in a catch block. If the exception is raised before the variable is defined, the catch block may have to handle an undefined variable, leading to more chaos.

```
<?php

$a = 1;
try {
    mayThrowAnException();
    $b = 2;
} catch (\Exception $e) {
    // $a is already defined, as it was done before the try block
    // $b may not be defined, as it was initialized after the exception-throwing␣
↪expression
    echo $a + $b;
}

?>
```

9.82.1 Suggestions

- Always define the variable used in the catch clause, before the try block.

Short name	Exceptions/CatchUndefinedVariable
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.83 Check All Types

When checking for time, avoid using else. Mention explicitly all tested type, and raise an exception when reaching else.

PHP has a short list of scalar types : null, boolean, integer, real, strings, object, resource and array. When a variable is not holding one the the type, then it may be of any other type.

Most of the time, when using a simple is_string() / else test, this is relying on the conception of the code. By construction, the arguments may be one of two types : array or string.

What happens often is that in case of failure in the code (database not working, another class not checking its results), a third type is pushed to the structure, and it ends up breaking the execution.

The safe way is to check the various types all the time, and use the default case (here, the else) to throw exception() or test an assertion and handle the special case.

```
<?php

// hasty version
if (is_array($argument)) {
    $out = $argument;
} else {
    // Here, $argument is NOT an array. What if it is an object ? or a NULL ?
    $out = array($argument);
}

// Safe type checking : do not assume that 'not an array' means that it is the other␣
↪expected type.
```

(continues on next page)

(continued from previous page)

```
if (is_array($argument)) {
    $out = $argument;
} elseif (is_string($argument)) {
    $out = array($argument);
} else {
    assert(false, '$argument is not an array nor a string, as expected!');
}

?>
```

Using is_callable(), is_iterable() with this structure is fine : when variable is callable or not, while a variable is an integer or else.

Using a type test without else is also accepted here. This is a special treatment for this test, and all others are ignored. This aspect may vary depending on situations and projects.

9.83.1 Suggestions

- Include a default case to handle all unknown situations
- Include and process explicit types as much as possible

Short name	Structures/CheckAllTypes
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Zend-Config*, *Vanilla*

9.84 Check Crypto Key Length

Each cryptography algorithm requires a reasonable length. Make sure an up-to-date length is used.

This rule use the following recommendations :

- OPENSSL_KEYTYPE_RSA’ => 3072
- OPENSSL_KEYTYPE_DSA’ => 2048
- OPENSSL_KEYTYPE_DH’ => 2048
- OPENSSL_KEYTYPE_EC’ => 512

The values above are used with the openssl PHP extension.

```
<?php

// Extracted from the documentation

// Generates a new and strong key
$private_key = openssl_pkey_new(array(
    private_key_type => OPENSSL_KEYTYPE_EC,
    private_key_bits => 1024,
));
```

(continues on next page)

(continued from previous page)

```
// Generates a new and weak key
$private_key = openssl_pkey_new(array(
    private_key_type => OPENSSL_KEYTYPE_EC,
    private_key_bits => 256,
));

?>
```

See also The Definitive 2019 Guide to Cryptographic Key Sizes and Algorithm Recommendations and Cryptographic Key Length Recommendation.

9.84.1 Suggestions

-

Short name	Security/CryptoKeyLength
Rulesets	*Security*
Severity	Minor
Time To Fix	Quick (30 mins)

9.85 Check JSON

Check errors whenever JSON is encoded or decoded.

In particular, `NULL` is a valid decoded JSON response. If you want to avoid mistaking NULL for an error, it is recommended to call `json_last_error`.

```
<?php

$encoded = json_encode($incoming);
// Unless JSON must contains some non-null data, this mistakes NULL and error
if(json_last_error() != JSON_ERROR_NONE) {
    die('Error when encoding JSON');
}

$decoded = json_decode($incoming);
// Unless JSON must contains some non-null data, this mistakes NULL and error
if($decoded === null) {
    die('ERROR');
}

?>
```

See also Option to make json_encode and json_decode throw exceptions on errors, json_last_error.

9.85.1 Suggestions

- Always check after JSON operation : encoding or decoding.
- Add a call to json_last_error()
- Configure operations to throw an exception upon error (`JSON_THROW_ON_ERROR`), and catch it.

Short name	Structures/CheckJson
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Woocommerce*

9.86 Check On __Call Usage

When using the magic methods __call() and __staticcall(), make sure the method exists before calling it.

If the method doesn’t exists, then the same method will be called again, leading to the same failure. Finally, it will crash PHP.

```
<?php

class safeCall {
    function __class($name, $args) {
        // unsafe call, no checks
        if (method_exists($this, $name)) {
            $this->$name(...$args);
        }
    }
}

class unsafeCall {
    function __class($name, $args) {
        // unsafe call, no checks
        $this->$name(...$args);
    }
}

?>
```

See also Method overloading and ‘‘Magical PHP: __call <https://www.garfieldtech.com/index.php/blog/magical-php-call>‘_.

9.86.1 Suggestions

- Add a call to method_exists() before using any method name
- Relay the call to another object that doesn’t handle __call() or __callStatic()

Short name	Classes/CheckOnCallUsage
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.87 Child Class Removes Typehint

PHP 7.2 introduced the ability to remove a typehint when overloading a method. This is not valid code for older versions.

```php
<?php

class foo {
    function foobar(foo $a) {}
}

class bar extends foo {
    function foobar($a) {}
}

?>
```

Short name	Classes/ChildRemoveTypehint
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *Typechecks*
Php Version	With PHP 7.2 and more recent
Severity	Major
Time To Fix	Quick (30 mins)

9.88 Class Const With Array

Constant defined with const keyword may be arrays but only stating with PHP 5.6. Define never accept arrays : it only accepts scalar values.

Short name	Php/ClassConstWithArray
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Php Version	With PHP 5.5 and more recent
Severity	Critical
Time To Fix	Slow (1 hour)

9.89 Class Could Be Final

Any class that has no extension should be `final` by default.

As stated by `Matthias Noback : If a class is not marked final, it has at least one subclass.`

Prevent your classes from being subclassed by making them `final`. Sometimes, classes are not meant or thought to be derivable.

```php
<?php

class x {}            // This class is extended
class y extends x {}  // This class is extended
class z extends y {}  // This class is not extended

final class z2 extends y {}  // This class is not extended
```

(continues on next page)

(continued from previous page)

```

?>
```

See also Negative architecture, and assumptions about code.

9.89.1 Suggestions

- Make the class final
- Extends the class

Short name	Classes/CouldBeFinal
Rulesets	*Analyze*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.90 Class Function Confusion

Avoid classes and functions bearing the same name.

When functions and classes bear the same name, calling them may be confusing. This may also lead to forgotten 'new' keyword.

```
<?php

class foo {}

function foo() {}

// Forgetting the 'new' operator is easy
$object = new foo();
$object = foo();

?>
```

9.90.1 Suggestions

- Use a naming convention to distinguish functions and classes
- Rename the class or the function (or both)
- Use an alias with a *use* expression

Short name	Php/ClassFunctionConfusion
Rulesets	*Semantics*
Severity	Minor
Time To Fix	Slow (1 hour)

9.91 Class Should Be Final By Ocramius

'Make your classes always final, if they implement an interface, and no other public methods are defined'.

When a class should be final, as explained by `Ocramius` (`Marco Pivetta`).

```
<?php

interface i1 {
    function i1() ;
}

// Class should final, as its public methods are in an interface
class finalClass implements i1 {
    // public interface
    function i1 () {}

    // private method
    private function a1 () {}
}

?>
```

See also When to declare classes final.

Short name	Classes/FinalByOcramius
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)

9.92 Class Without Parent

Classes should not refer to `parent` when it is not extending another class.

In PHP 7.4, it is a Deprecated warning. In PHP 7.3, it was a Fatal error, when the code was finally executed.

```
<?php

class x {
    function foo() {
        parent::foo();
    }
}
?>
```

9.92.1 Suggestions

- Update the class and make it extends another class
- Change the parent mention with a fully qualified name
- Remove the call to the parent altogether

Short name	Classes/NoParent
Rulesets	*Analyze*, *ClassReview*, *CI-checks*
Php Version	7.4-
Severity	Minor
Time To Fix	Quick (30 mins)

9.93 Class, Interface Or Trait With Identical Names

The following names are used at the same time for classes, interfaces or traits. For example,

```
<?php
    class a     { /* some definitions */ }
    interface a { /* some definitions */ }
    trait a     { /* some definitions */ }
?>
```

Even if they are in different namespaces, identical names makes classes easy to confuse. This is often solved by using alias at import time : this leads to more confusion, as a class suddenly changes its name.

Internally, PHP use the same list for all classes, interfaces and traits. As such, it is not allowed to have both a trait and a class with the same name.

In PHP 4, and PHP 5 before namespaces, it was not possible to have classes with the same name. They were simply included after a check.

9.93.1 Suggestions

- Use distinct names for every class, trait and interface.
- Keep eponymous classes, traits and interfaces in distinct files, for definition but also for usage. When this happens, rename one of them.

Short name	Classes/CitSameName
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*shopware*, *NextCloud*

9.94 Classes Mutually Extending Each Other

Those classes are extending each other, creating an extension loop. PHP will yield a fatal error at running time, even if it is compiling the code.

```
<?php

// This code is lintable but won't run
class Foo extends Bar { }
class Bar extends Foo { }

// The loop may be quite large
```

(continues on next page)

(continued from previous page)

```
class Foo extends Bar { }
class Bar extends Bar2 { }
class Bar2 extends Foo { }

?>
```

Short name	Classes/MutualExtension
Rulesets	*LintButWontExec*, *ClassReview*
Severity	Major
Time To Fix	Quick (30 mins)

9.95 Clone With Non-Object

The `clone` keyword must be used on variables, properties or results from a function or method call.

`clone` cannot be used with constants or literals.

```
<?php

class x { }
$x = new x();

// Valid clone
$y = clone $x;

// Invalid clone
$y = clone x;

?>
```

Cloning a non-object lint but won't execute.

See also Object cloning.

9.95.1 Suggestions

- Only clone containers (like variables, properties…)
- Add typehint to injected properties, so they are checked as objects.

Short name	Classes/CloneWithNonObject
Rulesets	*Analyze*, *LintButWontExec*
Severity	Minor
Time To Fix	Quick (30 mins)

9.96 Close Tags

PHP manual recommends that script should be left open, without the final closing ?>. This way, one will avoid the infamous bug 'Header already sent', associated with left-over spaces, that are lying after this closing tag.

Short name	Php/CloseTags
Rulesets	*Coding Conventions*
Severity	Minor
Time To Fix	Instant (5 mins)
ClearPHP	leave-last-closing-out

9.97 Closure Could Be A Callback

Closure or arrowfunction could be simplified to a callback. Callbacks are strings or arrays.

A simple closure that only returns arguments relayed to another function or method, could be reduced to a simpler expression. They

Closure may be simplified with a string, for functioncall, with an array for methodcalls and static methodcalls.

Performances : simplifying a closure tends to reduce the call time by 50%.

```php
<?php

// Simple and faster call to strtoupper
$filtered = array_map('strtoupper', $array);

// Here the closure doesn't add any feature over strtoupper
$filtered = array_map(function ($x) { return strtoupper($x);}, $array);

// Methodcall example : no fix
$filtered = array_map(function ($x) { return $x->strtoupper() ;}, $array);

// Methodcall example  : replace with array($y, 'strtoupper')
$filtered = array_map(function ($x) use ($y) { return $y->strtoupper($x) ;}, $array);

// Static methodcall example
$filtered = array_map(function ($x) { return $x::strtoupper() ;}, $array);

// Static methodcall example   : replace with array('A', 'strtoupper')
$filtered = array_map(function ($x) { return A::strtoupper($x) ;}, $array);

?>
```

See also Closure class and Callbacks / Callables.

9.97.1 Suggestions

- Replace the closure by a string, with the name of the called function
- Replace the closure by an array, with the name of the called method and the object as first element

Short name	Functions/Closure2String
Rulesets	*Suggestions*, *Performances*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Tine20*, *NextCloud*

9.98 Closure May Use $this

$this is automatically accessible to closures.

When closures were introduced in PHP, they couldn't use the $this variable, making is cumbersome to access local properties when the closure was created within an object.

```
<?php

// Invalid code in PHP 5.4 and less
class Test
{
    public function testing()
    {
        return function() {
            var_dump($this);
        };
    }
}

$object = new Test;
$function = $object->testing();
$function();

?>
```

This is not the case anymore since PHP 5.4.

See also Anonymous functions.

Short name	Php/ClosureThisSupport
Rulesets	*CompatibilityPHP53*
Php Version	With PHP 5.4 and older
Severity	Minor
Time To Fix	Quick (30 mins)

9.99 Coalesce And Concat

The concatenation operator dot has precedence over the coalesce operator ??.

```
<?php

// Parenthesis are the right solution when in doubt
echo a . ($b ?? 'd') . $e;

// 'a' . $b is evaluated first, leading ot a useless ?? operator
'a' . $b ?? $c;

// 'd' . 'e' is evaluated first, leading to $b OR 'de'.
echo $b ?? 'd' . 'e';

?>
```

9.99.1 Suggestions

- Add parenthesis around ?? operator to avoid misbehavior
- Do not use dot and ?? together in the same expression

Short name	Structures/CoalesceAndConcat
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.100 Coalesce Equal

Usage of coalesce assignement operator. The operator is available in PHP since PHP 7.4.

```
<?php

// Coalesce operator, since PHP 5.3
$a ??= 'default value';

// Equivalent to $a = $a ?? 'default value';

?>
```

See also Ternary Operator.

Short name	Php/CoalesceEqual
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.4 and more recent

9.101 Common Alternatives

In the following conditional structures, expressions were found that are common to both 'then' and 'else'. It may be interesting, though not always possible, to put them both out of the conditional, and reduce line count.

```
<?php
if ($c == 5) {
    $b = strtolower($b[2]);
    $a++;
} else {
    $b = strtolower($b[2]);
    $b++;
}
?>
```

may be rewritten in :

```
<?php

$b = strtolower($b[2]);
if ($c == 5) {
    $a++;
} else {
    $b++;
}

?>
```

9.101.1 Suggestions

- Collect common expressions, and move them before of after the if/then expression.
- Move a prefix and suffixes to a third-party method

Short name	Structures/CommonAlternatives
Rulesets	*Analyze*
Severity	Major
Time To Fix	Instant (5 mins)
Examples	*Dolibarr*, *NextCloud*

9.102 Compact Inexistant Variable

Compact() doesn't warn when it tries to work on an inexistant variable. It just ignores the variable.

This behavior changed in PHP 7.3, and compact() now emits a warning when the compacted variable doesn't exist.

```
<?php

function foo($b = 2) {
    $a = 1;
    // $c doesn't exists, and is not compacted.
    return compact('a', 'b', 'c');
}
?>
```

For performances reasons, this analysis only works inside methods and functions.

See also compact and PHP RFC: Make compact function reports undefined passed variables.

9.102.1 Suggestions

- Fix the name of variable in the compact() argument list
- Remove the name of variable in the compact() argument list

Short name	Php/CompactInexistant
Rulesets	*Suggestions*, *CompatibilityPHP73*
Severity	Major
Time To Fix	Quick (30 mins)

9.103 Compare Hash

When comparing hash values, it is important to use the strict comparison : hash_equals(), === or !==.

In a number of situations, the hash value will start with 0e, and PHP will understand that the comparison involves integers : it will then convert the strings into numbers, and it may end up converting them to 0.

Here is an example :

```
<?php

// The two following passwords hashes matches, while they are not the same.
$hashed_password = 0e462097431906509000000000000;
if (hash('md5','240610708',false) == $hashed_password) {
  print 'Matched.'.PHP_EOL;
}

// hash returns a string, that is mistaken with 0 by PHP
// The strength of the hashing algorithm is not a problem
if (hash('ripemd160','20583002034',false) == '0') {
  print 'Matched.'.PHP_EOL;
}

if (hash('md5','240610708',false) !== $hashed_password) {
  print 'NOT Matched.'.PHP_EOL;
}

// Display true
var_dump(md5('240610708') == md5('QNKCDZO') );

?>
```

You may also use password_hash() and password_verify() : they work together without integer conversion problems, and they can't be confused with a number.

See also Magic Hashes What is the best way to compare hashed strings? (PHP) and md5('240610708') == md5('QNKCDZO').

9.103.1 Suggestions

- Use dedicated functions for hash comparisons
- Use identity operators (===), and not equality operators (==) to compare hashes
- Compare hashes in the database (or external system), where such confusion is not possible

Short name	Security/CompareHash
Rulesets	*Security*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	strict-comparisons
Examples	*Traq*, *LiveZilla*

9.104 Compared Comparison

Usually, comparison are sufficient, and it is rare to have to compare the result of comparison. Check if this two-stage comparison is really needed.

```
<?php

if ($a === strpos($string, $needle) > 2) {}

// the expression above apply precedence :
// it is equivalent to :
if (($a === strpos($string, $needle)) > 2) {}

?>
```

See also Operators Precedence.

Short name	Structures/ComparedComparison
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)

9.105 Complex Dynamic Names

Avoid using expressions as names for variables or methods.

There are no place for checks or flow control, leading to any rogue value to be used as is. Besides, the expression is often overlooked, and not expected there : this makes the code less readable.

It is recommended to build the name in a separate variable, apply the usual checks for existence and validity, and then use the name.

```
<?php

$a = new foo();

// Code is more readable
$name = strolower($string);
if (!property_exists($a, $name)) {
    throw new missingPropertyexception($name);
}
echo $a->$name;

// This is not check
echo $a->{strtolower($string)};

?>
```

This analysis only accept simple containers, such as variables, properties.

See also Dynamically Access PHP Object Properties with ‘$this <https://drupalize.me/blog/201508/dynamically-access-php-object-properties>‘_.

9.105.1 Suggestions

- Extract the expression from the variable syntax, and make it a separate variable.

Short name	Variables/ComplexDynamicNames
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.106 Concat And Addition

Precedence between addition and concatenation has changed. In PHP 7.4, addition has precedence, and before, addition and concatenation had the same precedence.

From the RFC : `Currently the precedence of '.', '+' and '-' operators are equal. Any combination of these operators are simply evaluated left-to-right.`

This is counter-intuitive though: you rarely want to add or subtract concatenated strings which in general are not numbers. However, given PHP's capability of seamlessly converting an integer to a string, concatenation of these values is desired.‘‘

```
<?php
// Extracted from the RFC
echo sum: . $a + $b;

// current behavior: evaluated left-to-right
echo (sum: . $a) + $b;

// desired behavior: addition and subtraction have a higher precendence
echo sum : . ($a + $b);

?>
```

This analysis reports any addition and concatenation that are mixed, without parenthesis. Addition also means substraction here, aka using + or -.

The same applies to bitshift operations, << and >>. There is no RFC for this change.

See also Change the precedence of the concatenation operator.

9.106.1 Suggestions

- Add parenthesis around the addition to ensure its expected priority
- Move the addition outside the concatenation

Short name	Php/ConcatAndAddition
Rulesets	*Analyze*, *CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*, *CompatibilityPHP54*, *CompatibilityPHP74*, *CompatibilityPHP80*, *CompatibilityPHP55*, *CompatibilityPHP56*, *Top10*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.107 Concat Empty String

Using a concatenation to make a value a string should be replaced with a type cast.

Type cast to a string is done with `(string)` operator. There is also the function strval(), although it is less recommended.

```
<?php

$a = 3;

// explicite way to cast a value
$b = (string) $a; // $b is a string with the content 3

// Wrong way to cast a value
$c = $a . ''; // $c is a string with the content 3
$c = '' . $a; // $c is a string with the content 3
$a .= '';     // $a is a string with the content 3

// Wrong way to cast a value
$c = $a . '' . $b; // This is not reported. The empty string is useless, but not␣
↪meant to type cast

?>
```

See also Type Casting and PHP Type Casting.

9.107.1 Suggestions

- Avoid concatenating with empty strings
- Use (string) operator to cast to string
- Remove any concatenated empty string

Short name	Structures/ConcatEmpty
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.108 Concrete Visibility

Methods that implements an interface in a class must be public.

PHP does lint this, unless the interface and the class are in the same file. At execution, it stops immediately with a Fatal error : 'Access level to c::iPrivate() must be public (as in class i) ';

```
<?php

interface i {
    function iPrivate() ;
    function iProtected() ;
    function iPublic() ;
}

class c implements i {
    // Methods that implements an interface in a class must be public.
    private function iPrivate() {}
    protected function iProtected() {}
    public function iPublic() {}
}

?>
```

See also Interfaces.

9.108.1 Suggestions

- Always set interface methods to public.

Short name	Interfaces/ConcreteVisibility
Rulesets	*Analyze*, *LintButWontExec*
Severity	Major
Time To Fix	Instant (5 mins)

9.109 Configure Extract

The extract() function overwrites local variables when left unconfigured.

Extract imports variables from an array into the local scope. In case of a conflict, that is when a local variable already exists, it overwrites the previous variable.

In fact, extract() may be configured to handle the situation differently : it may skip the conflicting variable, prefix it, prefix it only if it exists, only import overwriting variables… It may also import them as references to the original values.

This analysis reports extract() when it is not configured explicitly. If overwriting is the intended objective, it is not reported.

```
<?php

// ignore overwriting variables
extract($array, EXTR_SKIP);
```

(continues on next page)

(continued from previous page)

```
// prefix all variables explicitly variables with 'php_'
extract($array, EXTR_PREFIX_ALL, 'php_');

// overwrites explicitly variables
extract($array, EXTR_OVERWRITE);

// overwrites implicitely variables : do we really want that?
extract($array, EXTR_OVERWRITE);

?>
```

Always avoid using extract() on untrusted sources, such as `$_GET`, `$_POST`, `$_FILES`, or even databases records.

See also extract.

9.109.1 Suggestions

- Always use the second argument of extract(), and avoid using `EXTR_OVERWRITE`

Short name	Security/ConfigureExtract
Rulesets	*Security*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Zurmo*, *Dolibarr*

9.110 Const Visibility Usage

Visibility for class constant controls the accessibility to class constant.

A public constant may be used anywhere in the code; a protected constant usage is restricted to the class and its relatives; a private constant is restricted to itself.

This feature was introduced in PHP 7.1. It is recommended to use explicit visibility, and, whenever possible, make the visibility private.

```
<?php

class x {
    public const a = 1;
    protected const b = 2;
    private const c = 3;
    const d = 4;
}

interface i {
    public const a = 1;
      const d = 4;
}

?>
```

See also Class Constants and PHP RFC: Support Class Constant Visibility.

9.110.1 Suggestions

- Add constant visibility, at least ‘public’.

Short name	Classes/ConstVisibilityUsage
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.1 and more recent
Severity	Minor
Time To Fix	Slow (1 hour)

9.111 Const With Array

The const keyword supports array. This feature was added in PHP 5.6.

The array must be filled with other constants. It may also be build using the ‘+’ operator.

```
<?php

const PRIMES = [2, 3, 5, 7];

class X {
    const TWENTY_THREE = 23;
    const MORE_PRIMES = PRIMES + [11, 13, 17, 19];
    const EVEN_MORE_PRIMES = self::MORE_PRIMES + [self::TWENTY_THREE];
}

?>
```

See also Class Constants and Constants Syntax.

Short name	Php/ConstWithArray
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Php Version	With PHP 5.5 and more recent
Severity	Major
Time To Fix	Slow (1 hour)

9.112 Constant Class

A class or an interface only made up of constants. Constants usually have to be used in conjunction of some behavior (methods, class…) and never alone.

```
<?php

class ConstantClass {
    const KBIT = 1000;
    const MBIT = self::KBIT * 1000;
    const GBIT = self::MBIT * 1000;
```

(continues on next page)

(continued from previous page)

```
    const PBIT = self::GBIT * 1000;
}

?>
```

As such, they should be PHP constants (build with define or const), or included in a class with other methods and properties.

See also PHP Classes containing only constants.

9.112.1 Suggestions

- Make the class an interface
- Make the class an abstract class, to avoid its instantiation

Short name	Classes/ConstantClass
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Slow (1 hour)

9.113 Constant Comparison

Constant to the left or right is a favorite.

Comparisons are commutative : they may be $a == B or B == $a. The analyzed code show less than 10% of one of the two : for consistency reasons, it is recommended to make them all the same.

Putting the constant on the left is also called 'Yoda Comparison', as it mimics the famous characters style of speech. It prevents errors like 'B = $a' where the comparison is turned into an assignation.

The natural way is to put the constant on the right. It is often less surprising.

Every comparison operator is used when finding the favorite.

```
<?php

//
if ($a === B) { doSomething(); }
if ($c > D) { doSomething(); }
if ($e !== G) { doSomething(); }
do { doSomething(); } while ($f === B);
while ($a === B) { doSomething(); }

// be consistent
if (B === $a) {}

// Compari
if (B <= $a) {}

?>
```

Short name	Structures/ConstantComparisonConsistance
Rulesets	*Coding Conventions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.114 Constant Scalar Expressions

Define constant with the result of static expressions. This means that constants may be defined with the const keyword, with the help of various operators but without any functioncalls.

This feature was introduced in PHP 5.6. It also supports array(), and expressions in arrays.

Those expressions (using simple operators) may only manipulate other constants, and all values must be known at compile time.

```php
<?php

// simple definition
const A = 1;

// constant scalar expression
const B = A * 3;

// constant scalar expression
const C = [A ** 3, '3' => B];

?>
```

See also Constant Scalar Expressions.

Short name	Structures/ConstantScalarExpression
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Php Version	With PHP 5.6 and more recent
Severity	Major
Time To Fix	Quick (30 mins)

9.115 Constants Created Outside Its Namespace

Constants Created Outside Its Namespace.

Using the define() function, it is possible to create constant outside their namespace, but using the fully qualified namespace.

```php
<?php

namespace A\B {
    // define A\B\C as 1
    define('C', 1);
}

namespace D\E {
    // define A\B\C as 1, while outside the A\B namespace
```

(continues on next page)

(continued from previous page)

```
    define('A\B\C', 1);
}

?>
```

However, this makes the code confusing and difficult to debug. It is recommended to move the constant definition to its namespace.

Short name	Constants/CreatedOutsideItsNamespace
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)

9.116 Constants With Strange Names

List of constants being defined with names that are incompatible with PHP standards.

```
<?php

// Define a valid PHP constant
define('ABC', 1);
const ABCD = 2;

// Define an invalid PHP constant
define('ABC!', 1);
echo defined('ABC!') ? constant('ABC!') : 'Undefined';

// Const doesn't allow illegal names

?>
```

See also PHP Constants.

Short name	Constants/ConstantStrangeNames
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Slow (1 hour)

9.117 Continue Is For Loop

break and continue are very similar in PHP : they both break out of loop or switch. Yet, continue should be reserved for loops.

Since PHP 7.3, the execution will emit a warning when finding a continue inside a switch inside a loop : '"continue" targeting switch is equivalent to "break". Did you mean to use "continue 2"?'

```
<?php

while ($foo) {
    switch ($bar) {
```

(continues on next page)

(continued from previous page)

```
        case 'baz':
            continue; // In PHP: Behaves like 'break;'
                      // In C:   Behaves like 'continue 2;'
    }
}

?>
```

See also Deprecate and remove ‘continue targeting switch <https://wiki.php.net/rfc/continue_on_switch_deprecation>‘_.

9.117.1 Suggestions

- Replace break by continue

Short name	Structures/ContinueIsForLoop
Rulesets	*Analyze*, *CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP73*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*XOOPS*

9.118 Could Be Abstract Class

An abstract class is never instantiated, and has children class that are. As such, a ‘parent’ class that is never instantiated by itself, but has its own children instantiated could be marked as abstract.

That will prevent new code to try to instantiate it.

```
<?php

// Example code would actually be split over multiple files.


// That class could be abstract
class motherClass {}

// Those classes shouldn't be abstract
class firstChildren extends motherClass {}
class secondChildren extends motherClass {}
class thirdChildren extends motherClass {}

new firstChildren();
new secondChildren();
new thirdChildren();

//Not a single : new motherClass()
```

(continues on next page)

(continued from previous page)

```
?>
```

See also Class Abstraction Abstract classes and methods.

9.118.1 Suggestions

- Make this class an abstract class

Short name	Classes/CouldBeAbstractClass
Rulesets	*Analyze*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Edusoho*, *shopware*

9.119 Could Be Callable

Mark arguments and return types that can be set to `callable`.

```
<?php

// Accept a callable as input
function foo($b) {
    // Returns value as return
    return $b();
}

?>
```

9.119.1 Suggestions

- Add *callable* typehint to the code.

Short name	Typehints/CouldBeCallable
Rulesets	*Typechecks*
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

9.120 Could Be Class Constant

When a property is defined and read, but never modified, it may be a constant.

```
<?php

class foo {
    // $this->bar is never modified.
```

(continues on next page)

(continued from previous page)

```
    private $bar = 1;

    // $this->foofoo is modified, at least once
    private $foofoo = 2;

    function method($a) {
        $this->foofoo = $this->bar + $a + $this->foofoo;

        return $this->foofoo;
    }

}

?>
```

Starting with PHP 5.6, even array() may be defined as constants.

Short name	Classes/CouldBeClassConstant
Rulesets	*ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.121 Could Be Constant

Literals may be replaced by an existing constant.

Constants makes the code easier to read, as they may bear a meaningful name. They also hide implementation values, with a readable name, such as `const READABLE= true;`. Later, upgrading constant values is easier than scouring the code with a new literal.

Not all literal can be replaced by a constant values : sometimes, literal may have the same literal value, but different meanings. Check with your application semantics before changing any literal with a constant.

```
<?php

const A = 'abc';
define('B', 'ab');

class foo {
    const X = 'abcd';
}

// Could be replaced by B;
$a = 'ab';

// Could be replaced by A;
$a = 'abc';

// Could be replaced by foo::X;
$a = 'abcd';

?>
```

This analysis currently doesn’t support arrays.

This analysis also skips very common values, such as boolean, `0` and `1`. This prevents too many false positive.

9.121.1 Suggestions

- Turn the literal into an existing constant

Short name	Constants/CouldBeConstant
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.122 Could Be Else

Merge opposition conditions into one if/then structure.

When two if/then structures follow each other, using a condition and its opposite, they may be merged into one.

```php
<?php

// Short version
if ($a == 1) {
    $b = 2;
} else {
    $b = 1;
}

// Long version
if ($a == 1) {
    $b = 2;
}

if ($a != 1) {
    $b = 3;
}

?>
```

9.122.1 Suggestions

- Merge the two conditions into one structure
- Check if the second condition is still applicable

Short name	Structures/CouldBeElse
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*SugarCrm*, *OpenEMR*

9.123 Could Be Float

Mark arguments, properties and return types that can be set to `float`.

```
<?php

// Accept an int as input
function foo($b) {
    // Returns a float (cubic root of $b);
    return pow($b, 1 / 3);
}

?>
```

9.123.1 Suggestions

- Add *float* typehint to the code.

Short name	Typehints/CouldBeFloat
Rulesets	*Typechecks*
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

9.124 Could Be Integer

Mark arguments, properties and return types that can be set to `int`.

```
<?php

// Accept an int as input
function foo($b) {
    // Returns an int
    return $b + 8;
}

?>
```

9.124.1 Suggestions

- Add *int* typehint to the code.

Short name	Typehints/CouldBeInt
Rulesets	*Typechecks*
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

9.125 Could Be Iterable

Mark arguments, properties and return types that can be set to `iterable`.

```
<?php

// Accept an array or a traversable Object as input
function foo($b) {
    foreach($b as $c) {

    }

    // Returns an array
    return [$b];
}

?>
```

9.125.1 Suggestions

- Add *iterable* typehint to the code (PHP 8.0+).

Short name	Typehints/CouldBeIterable
Rulesets	*Typechecks*
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

9.126 Could Be Null

Mark arguments and return types that can be null.

```
<?php

// Accept null as input, when used as third argument of file_get_contents
function foo($b) {
    $s = file_get_contents(URL, false, $b);

    // Returns a string
    return shell_exec($s);
}

?>
```

9.126.1 Suggestions

- Add *null* typehint to the code (PHP 8.0+).
- Add *?* typehint to the code.

Short name	Typehints/CouldBeNull
Rulesets	*Typechecks*
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

9.127 Could Be Parent

Mark arguments, return types and properties that can be set to `parent`.

This analysis works when typehints have already been configured.

```
<?php

class x extends w {
    // Accept a w object as input
    function foo(w $b) : w {
        // Returns a w object
        return $b;
    }
}

?>
```

9.127.1 Suggestions

- Add *parent* typehint to the code.
- Add the literal class/type typehint to the code.

Short name	Typehints/CouldBeParent
Rulesets	*Typechecks*
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

9.128 Could Be Parent Method

A method is defined in several children, but not in a the parent class. It may be worth checking if this method doesn't belong the parent class, as an abstraction.

```
<?php

// The parent class
class x { }

// The children class
class y1 extends x {
    // foo is common to y1 and y2, so it shall be also a method in x
    function foo() {}
```

(continues on next page)

(continued from previous page)

```
    // fooY1 is specific to y1
    function fooY1() {}
}

class y2 extends x {
    function foo() {}
    // fooY2 is specific to y1
    function fooY2() {}
}

?>
```

Only the name of the method is used is for gathering purposes. If the code has grown organically, the signature (default values, typehint, argument names) may have followed different path, and will require a refactorisation.

9.128.1 Suggestions

- Create an abstract method in the parent
- Create an concrete method in the parent, and move default behavior there by removing it in children classes

Name	Default	Type	Description
minChildren	4	integer	Minimal number of children using this method.

Short name	Classes/CouldBeParentMethod
Rulesets	*ClassReview*
Severity	Minor
Time To Fix	Slow (1 hour)

9.129 Could Be Private Class Constant

Class constant may use `private` visibility.

Since PHP 7.1, constants may also have a public/protected/private visibility. This restrict their usage to anywhere, class and children or class.

As a general rule, it is recommended to make constant `private` by default, and to relax this restriction as needed. PHP makes them public by default.

```
<?php

class foo {
    // pre-7.1 style
    const PRE_71_CONSTANT = 1;

    // post-7.1 style
    private const PRIVATE_CONSTANT = 2;
    public const PUBLIC_CONSTANT = 3;

    function bar() {
        // PRIVATE CONSTANT may only be used in its class
```

(continues on next page)

(continued from previous page)

```
        echo self::PRIVATE_CONSTANT;
    }
}

// Other constants may be used anywhere
function x($a = foo::PUBLIC_CONSTANT) {
    echo $a.' '.foo:PRE_71_CONSTANT;
}

?>
```

Constant shall stay `public` when the code has to be compatible with PHP 7.0 and older.

They also have to be public in the case of component : some of those constants have to be used by external actors, in order to configure the component.

See also Class Constants.

Short name	Classes/CouldBePrivateConstante
Rulesets	*ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Phinx*

9.130 Could Be Protected Class Constant

Class constant may use 'protected' visibility.

Since PHP 7.1, constants may also have a public/protected/private visibility. This restrict their usage to anywhere, class and children or class.

As a general rule, it is recommended to make constant 'private' by default, and to relax this restriction as needed. PHP makes them public by default.

```
<?php

class foo {
    // pre-7.1 style
    const PRE_71_CONSTANT = 1;

    // post-7.1 style
    protected const PROTECTED_CONSTANT = 2;
    public const PUBLIC_CONSTANT = 3;
}

class foo2 extends foo {
    function bar() {
        // PROTECTED_CONSTANT may only be used in its class or its children
        echo self::PROTECTED_CONSTANT;
    }
}

class foo3 extends foo {
    function bar() {
```

(continues on next page)

(continued from previous page)

```
        // PROTECTED_CONSTANT may only be used in its class or any of its children
        echo self::PROTECTED_CONSTANT;
    }
}

// Other constants may be used anywhere
function x($a = foo::PUBLIC_CONSTANT) {
    echo $a.' '.foo:PRE_71_CONSTANT;
}

?>
```

Short name	Classes/CouldBeProtectedConstant
Rulesets	*ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.131 Could Be Protected Method

Those methods are declared public, but are never used publicly. They may be made protected.

```
<?php

class foo {
    // Public, and used publicly
    public publicMethod() {}

    // Public, but never used outside the class or its children
    public protectedMethod() {}

    private function bar() {
        $this->protectedMethod();
    }
}

$foo = new Foo();
$foo->publicMethod();

?>
```

These properties may even be made private.

Short name	Classes/CouldBeProtectedMethod
Rulesets	*ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.132 Could Be Protected Property

Those properties are declared public, but are never used publicly. They may be made protected.

```php
<?php

class foo {
    // Public, and used publicly
    public $publicProperty;
    // Public, but never used outside the class or its children
    public $protectedProperty;

    function bar() {
        $this->protectedProperty = 1;
    }
}

$foo = new Foo();
$foo->publicProperty = 3;

?>
```

This property may even be made private.

Short name	Classes/CouldBeProtectedProperty
Rulesets	*ClassReview*
Severity	Minor
Time To Fix	Slow (1 hour)

9.133 Could Be Self

Mark arguments, return types and properties that can be set to `self`.

This analysis works when typehints have already been configured.

```php
<?php

class x {
    // Accept a x object as input
    function foo(x $b) : x {
        // Returns a x object
        return $b;
    }
}

?>
```

9.133.1 Suggestions

- Add *self* typehint to the code.
- Add the literal class/type typehint to the code.

Short name	Typehints/CouldBeSelf
Rulesets	*Typechecks*
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

9.134 Could Be Static

This global is only used in one function or method. It may be called 'static', instead of global. This allows you to keep the value between call to the function, but will not be accessible outside this function.

```
<?php
function foo( ) {
    static $variableIsReservedForX; // only accessible within foo( ), even between␣
↪calls.
    global $variableIsGlobal;       //      accessible everywhere in the application
}
?>
```

Short name	Structures/CouldBeStatic
Rulesets	*Analyze*, *ClassReview*, *Analyze*, *ClassReview*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Dolphin*, *Contao*

9.135 Could Be Static Closure

Closure may be static, and prevent the import of `$this`.

By preventing the useless import of `$this`, you avoid useless work.

This also has the added value to prevent the usage of `$this` from the closure. This is a good security practice.

```
<?php

class Foo
{
    function __construct()
    {

        // Not possible to use $this
        $func = static function() {
            var_dump($this);
        };
        $func();

        // Normal import of $this
        $closure = function() {
            var_dump($this);
        };
    }
```

(continues on next page)

(continued from previous page)

```
};
new Foo();

?>
```

This is a micro-optimisation. Apply it in case of intensive usage.

See also Anonymous functions, GeneratedHydrator and Static anonymous functions <https://www.php.net/manual/en/functions.anonymous.php#functions.anonymous-functions.'static>'_.

9.135.1 Suggestions

- Add the static keyword to the closure.
- Make actual usage of $this in the closure.

Short name	Functions/CouldBeStaticClosure
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Piwigo*

9.136 Could Be String

Mark arguments and return types that can be set to string.

```
<?php

// Accept a string as input
function foo($a) {
    // Returns a string
    return $a . 'string';
}

?>
```

9.136.1 Suggestions

- Choose the string typehint, and add it to the code.

Short name	Typehints/CouldBeString
Rulesets	*Typechecks*
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

9.137 Could Be Stringable

Stringable is an interface that mark classes as string-castable. It is introduced in PHP 8.0.

Classes that defined a __toString() magic method may be turned into a string when the typehint, argument, return or property, requires it. This is not the case when strict_types is activated. Yet, until PHP 8.0, there was nothing to identify a class as such.

```php
<?php

// This class may implement Stringable
class x {
    function __tostring() {
        return 'asd';
    }
}

echo (new x);

?>
```

See also PHP RFC: Add Stringable interface.

9.137.1 Suggestions

-

Short name	Classes/CouldBeStringable
Rulesets	*Analyze*, *LintButWontExec*
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)

9.138 Could Be Void

Mark return types that can be set to void.

```php
<?php

// No return, this should be void.
function foo() {
    ++$a; // Not useful
}

?>
```

All abstract methods (in classes or in interfaces) are omitted here.

9.138.1 Suggestions

- Add the void typehint to the code.

Short name	Typehints/CouldBeVoid
Rulesets	*Typechecks*
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

9.139 Could Make A Function

When a function is called across the code with the same arguments often enough, it should be turned into a local API.

This approach is similar to turning literals into constants : it centralize the value, it helps refactoring by updating it. It also makes the code more readable. Moreover, it often highlight common grounds between remote code locations.

The analysis looks for functions calls, and checks the arguments. When the calls occurs more than 4 times, it is reported.

```php
<?php

// str_replace is used to clean '&' from strings.
// It should be upgraded to a central function
function foo($arg ) {
    $arg = str_replace('&', '', $arg);
    // do something with $arg
}

class y {
    function bar($database ) {
        $value = $database->queryName();
        $value = str_replace('&', '', $value);
        // $value = removeAmpersand($value);
        // do something with $arg2
    }
}

// helper function
function removeAmpersand($string) {
    return str_replace('&', '', $string);
}

?>
```

See also Don’t repeat yourself (DRY).

9.139.1 Suggestions

- Create a constant for common pieces of data
- Create a function based on context-free repeated elements
- Create a class based on repeated elements with dependent values

Name	Default	Type	Description
centralizeThreshold	8	integer	Minimal number of calls of the function with one common argument.

Short name	Functions/CouldCentralize
Rulesets	*Analyze*, *Suggestions*
Severity	Minor
Time To Fix	Slow (1 hour)

9.140 Could Use Alias

This long name may be reduced by using an available alias.

This applies to classes (as full name or prefix), and to constants and functions.

```
<?php

use a\b\c;
use function a\b\c\foo;
use const a\b\c\D;

// This may be reduced with the above alias to c\d()
new a\b\c\d();

// This may be reduced to c\d\e\f
new a\b\c\d\e\f();

// This may be reduced to c()
new a\b\c();

// This may be reduced to D
echo a\b\c\D;

// This may be reduced to D
a\b\c\foo();

// This can't be reduced : it is an absolute name
\a\b\c\foo();

// This can't be reduced : it is no an alias nor a prefix
a\b\d\foo();

?>
```

9.140.1 Suggestions

- Use all your aliases so as to make the code shorter and more readable
- Add new aliases for missing path
- Make class names absolute and drop the aliases

Short name	Namespaces/CouldUseAlias
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.141 Could Use Compact

Compact() turns a group of variables into an array. It may be used to simplify expressions.

```
<?php

$a = 1;
$b = 2;

// Compact call
$array = compact('a', 'b');

$array === [1, 2];

// Detailing all the keys and their value
$array = ['a' => $a, 'b' => $b];

?>
```

Note that compact accepts any string, and any undefined variable is not set, without a warning.

See also compact.

9.141.1 Suggestions

- Replace the array() call with a compact() call.

Short name	Structures/CouldUseCompact
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*WordPress*

9.142 Could Use Promoted Properties

Promoted properties reduce PHP code at __construct() time. This feature is available in PHP 8.0.

```
<?php

class x {
    function __construct($a, $b) {
        // $a argument may be promoted to property $c
        $this->c = $a;

        // $b argument cannot be upgraded to property, as it is updated.
        // Move the addition to the new call, or keep the syntax below
        $this->d = $b + 2;
    }
}

?>
```

See also PHP 8: Constructor property promotion and PHP RFC: Constructor Property Promotion.

9.142.1 Suggestions

- Update the constructor syntax, and remove the property specification.

Short name	Php/CouldUsePromotedProperties
Rulesets	*Suggestions*
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)

9.143 Could Use Short Assignation

Use short assignment operator, to speed up code, and keep syntax clear.

Some operators, like * or +, have a compact and fast 'do-and-assign' version. They looks like a compacted version for = and the operator. This syntax is good for readability, and saves some memory in the process.

Depending on the operator, not all permutations of arguments are possible.

Addition and short assignation of addition have a different set of features when applied to arrays. Do not exchange one another in that case.

```php
<?php

$a = 10 + $a;
$a += 10;

$b = $b - 1;
$b -= 1;

$c = $c * 2;
$c *= 2;

$d = $d / 3;
$d /= 3;

$e = $e % 4;
$e %= 4;

$f = $f | 5;
$f |= 5;

$g = $g & 6;
$g &= 6;

$h = $h ^ 7;
$h ^= 7;

$i = $i >> 8;
$i >>= 8;

$j = $j << 9;
$j <<= 9;

// PHP 7.4 and more recent
```

(continues on next page)

(continued from previous page)

```
$l = $l ?? 'value';
$l ??= 'value';

?>
```

Short operators are faster than the extended version, though it is a micro-optimization.

See also Assignation Operators.

9.143.1 Suggestions

- Change the expression to use the short assignation

Short name	Structures/CouldUseShortAssignation
Rulesets	*Analyze*, *Performances*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
ClearPHP	use-short-assignations
Examples	*ChurchCRM*, *Thelia*

9.144 Could Use Try

Some commands may raise exceptions. It is recommended to use the try/catch block to intercept those exceptions, and process them.

- / : `DivisionByZeroError`
- % : `DivisionByZeroError`
- intdiv() : `DivisionByZeroError`
- << : `ArithmeticError`
- >> : `ArithmeticError`
- `Phar\:\:mungserver` : `PharException`
- `Phar\:\:webphar` : `PharException`

See also Predefined Exceptions, PharException.

9.144.1 Suggestions

- Add a try/catch clause around those commands
- Add a check on the values used with those operator : for example, check a dividend is not 0, or a bitshift is not negative

Short name	Exceptions/CouldUseTry
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Mautic*

9.145 Could Use __DIR__

Use __DIR__ constant to access the current file's parent directory.

Avoid using dirname() on __FILE__.

```
<?php

// Better way
$fp = fopen(__DIR__.'/myfile.txt', 'r');

// compatible, but slow way
$fp = fopen(dirname(__FILE__).'/myfile.txt', 'r');

// Since PHP 5.3
assert(dirname(__FILE__) == __DIR__);

?>
```

__DIR__ has been introduced in PHP 5.3.0.

See also Magic Constants.

9.145.1 Suggestions

- Use __DIR__ instead of `dirname(__FILE__);`

Short name	Structures/CouldUseDir
Rulesets	*Analyze*, *Suggestions*, *php-cs-fixable*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Woocommerce*, *Piwigo*

9.146 Could Use array_fill_keys

array_fill_keys() is a native PHP function that creates an array from keys. It gets the list of keys, and a constant value to assign to each keys.

This is twice faster than doing the same with a loop.

Note that is possible to use an object as initializing value : every element of the final array will be pointing to the same value. And, also, using an object as initializing value means that the same object will be used for each key : the object will not be cloned for each value.

```
<?php

$array = range('a', 'z');

// Fast way to build the array
$b = array_fill_keys($a, 0);

// Fast way to build the array, but every element will be the same object
$b = array_fill_keys($a, new Stdclass());
```

(continues on next page)

(continued from previous page)

```

// Slow way to build the array
foreach($array as $a) {
    $b[$a] = 0;
}

// Setting everything to null, slowly
$array = array_map(function() {}, $array);

?>
```

See also array_fill_keys.

9.146.1 Suggestions

- Use array_fill_keys()

Short name	Structures/CouldUseArrayFillKeys
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*ChurchCRM*, *PhpIPAM*

9.147 Could Use array_unique

Use array_unique() to collect unique elements from an array.

Always try to use native PHP functions, instead of rebuilding them with custom PHP code.

```
<?php

    $unique = array();
    foreach ($array as $b) {
        if (!in_array($b, $unique)) {
            /*  May be more code */
            $unique[] = $b;
        }
    }
?>
```

See also array_unique.

9.147.1 Suggestions

- Turn the foreach() and its condition into a call to array_unique()
- Extract the condition from the foreach() and add a separate call to array_unique()

Short name	Structures/CouldUseArrayUnique
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Dolibarr*, *OpenEMR*

9.148 Could Use self

`self` keyword refers to the current class, or any of its parents. Using it is just as fast as the full class name, it is as readable and it is will not be changed upon class or namespace change.

It is also routinely used in traits : there, `self` represents the class in which the trait is used, or the trait itself.

```
<?php

class x {
    const FOO = 1;

    public function bar() {
        return self::FOO;
// same as return x::FOO;
    }
}

?>
```

See also Scope Resolution Operator (::).

9.148.1 Suggestions

- replace the explicit name with self

Short name	Classes/ShouldUseSelf
Rulesets	*Analyze*, *Suggestions*, *ClassReview*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*WordPress*, *LiveZilla*

9.149 Could Use str_repeat()

Use str_repeat() or str_pad() instead of making a loop.

Making a loop to repeat the same concatenation is actually much longer than using str_repeat(). As soon as the loop repeats more than twice, str_repeat() is much faster. With arrays of 30, the difference is significant, though the whole operation is short by itself.

```
<?php

// This adds 7 'e' to $x
$x .= str_repeat('e', 7);
```

(continues on next page)

(continued from previous page)

```
// This is the same as above,
for($a = 3; $a < 10; ++$a) {
    $x .= 'e';
}

// here, $default must contains 7 elements to be equivalent to the previous code
foreach($default as $c) {
    $x .= 'e';
}

?>
```

9.149.1 Suggestions

- Use strrepeat() whenever possible

Short name	Structures/CouldUseStrrepeat
Rulesets	*Analyze*, *Top10*, *CI-checks*
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high
Examples	*Zencart*

9.150 Crc32() Might Be Negative

crc32() may return a negative number, on 32 bits platforms.

According to the manual : Because PHP's integer type is signed many CRC32 checksums will result in negative integers on 32 bits platforms. On 64 bits installations, all crc32() results will be positive integers though.

```
<?php

// display the checksum with %u, to make it unsigned
echo sprintf('%u', crc32($str));

// turn the checksum into an unsigned hexadecimal
echo dechex(crc32($str));

// avoid concatenating crc32 to a string, as it may be negative on 32bits platforms
echo 'prefix'.crc32($str);

?>
```

See also crc32().

Short name	Php/Crc32MightBeNegative
Rulesets	*Analyze*
Severity	Major
Time To Fix	Slow (1 hour)

9.151 Cyclic References

Avoid cyclic references.

Cyclic references happen when an object points to another object, which reciprocate. This is particularly possible with classes, when the child class has to keep a reference to the parent class.

```
<?php

class a {
    private $p = null;

    function foo() {
        $this->p = new b();
        // the current class is stored in the child class
        $this->p->m($this);
    }
}

class b {
    private $pb = null;

    function n($a) {
        // the current class keeps a link to its parent
        $this->pb = $a;
    }
}
?>
```

Cyclic references, or circular references, are memory intensive : only the garbage collector can understand when they may be flushed from memory, which is a costly operation. On the other hand, in an acyclic reference code, the reference counter will know immediately know that an object is free or not.

See also About circular references in PHP and A Journey to find a memory leak.

9.151.1 Suggestions

- Use a different object when calling the child objects.
- Refactor your code to avoid the cyclic reference.

Short name	Classes/CyclicReferences
Rulesets	*Analyze*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.152 Dangling Array References

Always unset a referenced-variable used in a loop.

It is highly recommended to unset blind variables when they are set up as references after a loop.

```
<?php

$array = array(1,2,3,4);

foreach($array as &$a) {
    $a += 1;
}
// This only unset the reference, not the value
unset($a);


// Dangling array problem
foreach($array as &$a) {
    $a += 1;
}
//$array === array(3,4,5,6);

// This does nothing (apparently)
// $a is already a reference, even if it doesn't show here.
foreach($array as $a) {}
//$array === array(3,4,5,5);

?>
```

When omitting this step, the next loop that will also require this variable will deal with garbage values, and produce unexpected results.

See also : No Dangling Reference, PHP foreach pass-by-reference: Do it right, or better not at all, How does PHP 'foreach' actually work?, References and foreach.

9.152.1 Suggestions

- Avoid using the reference altogether : sometimes, the reference is not needed.
- Add unset() right after the loop, to avoid reusing the reference.

Short name	Structures/DanglingArrayReferences
Rulesets	*Analyze*, *Top10*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)
ClearPHP	no-dangling-reference
Examples	*Typo3*, *SugarCrm*

9.153 Deep Definitions

Structures, such as functions, classes, interfaces, traits, etc. may be defined anywhere in the code, including inside functions. This is legit code for PHP.

Since the availability of autoload, with spl_register_autoload(), there is no need for that kind of code. Structures should be defined, and accessible to the autoloading. Inclusions and deep definitions should be avoided, as they compel code to load some definitions, while autoloading will only load them if needed.

```php
<?php

class X {
    function init() {
        // myFunction is defined when and only if X::init() is called.
        if (!function_exists('myFunction'){
            function myFunction($a) {
                return $a + 1;
            }
        })
    }
}

?>
```

Functions are excluded from autoload, but shall be gathered in libraries, and not hidden inside other code.

Constants definitions are tolerated inside functions : they may be used for avoiding repeat, or noting the usage of such function.

Definitions inside a if/then statement, that include PHP version check are accepted here.

See also Autoloading Classes.

9.153.1 Suggestions

- Move function definitions to the global space : outside structures, and method.

Short name	Functions/DeepDefinitions
Rulesets	*Analyze*
Severity	Major
Time To Fix	Slow (1 hour)
Examples	*Dolphin*

9.154 Define With Array

PHP 7.0 has the ability to define an array as a constant, using the define() native call. This was not possible until that version, only with the const keyword.

```php
<?php

//Defining an array as a constant
define('MY_PRIMES', [2, 3, 5, 7, 11]);

?>
```

Short name	Php/DefineWithArray
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and more recent
Severity	Critical
Time To Fix	Slow (1 hour)

9.155 Dependant Abstract Classes

Abstract classes should be autonomous. It is recommended to avoid depending on methods, constant or properties that should be made available in inheriting classes, without explicitly abstracting them.

The following abstract classes make usage of constant, methods and properties, static or not, that are not defined in the class. This means the inheriting classes must provide those constants, methods and properties, but there is no way to enforce this.

This may also lead to dead code : when the abstract class is removed, the host class have unused properties and methods.

```
<?php

// autonomous abstract class : all it needs is within the class
abstract class c {
    private $p = 0;

    function foo() {
        return ++$this->p;
    }
}

// dependant abstract class : the inheriting classes needs to provide some properties␣
↪or methods
abstract class c2 {
    function foo() {
        // $p must be provided by the extending class
        return ++$this->p;
    }
}

class c3 extends c2 {
    private $p = 0;
}
?>
```

See also *Dependant Trait*.

9.155.1 Suggestions

- Make the class only use its own resources
- Split the class in autonomous classes
- Add local property definitions to make the class independent

Short name	Classes/DependantAbstractClass
Rulesets	*Analyze*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.156 Dependant Trait

Traits should be autonomous. It is recommended to avoid depending on methods or properties that should be in the using class.

The following traits make usage of methods and properties, static or not, that are not defined in the trait. This means the host class must provide those methods and properties, but there is no way to enforce this.

This may also lead to dead code : when the trait is removed, the host class have unused properties and methods.

```
<?php

// autonomous trait : all it needs is within the trait
trait t {
    private $p = 0;

    function foo() {
        return ++$this->p;
    }
}

// dependant trait : the host class needs to provide some properties or methods
trait t2 {
    function foo() {
        return ++$this->p;
    }
}

class x {
    use t2;

    private $p = 0;
}
?>
```

See also *Dependant Abstract Classes*.

9.156.1 Suggestions

- Add local property definitions to make the trait independent
- Make the trait only use its own resources
- Split the trait in autonomous traits

Short name	Traits/DependantTrait
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*Zencart*

9.157 Deprecated Functions

The following functions are deprecated. It is recommended to stop using them now and replace them with a durable equivalent.

Note that these functions may be still usable : they generate warning that help tracking their usage in the log. To eradicate their usage, watch the logs, and update any deprecated warning. This way, the code won't be stuck when the function is actually removed from PHP.

```
<?php

// This is the current function
list($day, $month, $year) = explode('/', '08/06/1995');

// This is deprecated
list($day, $month, $year) = split('/', '08/06/1995');

?>
```

9.157.1 Suggestions

- Replace those deprecated with modern syntax
- Stop using deprecated syntax

Short name	Php/Deprecated
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	no-deprecated
Examples	*Dolphin*

9.158 Dereferencing String And Arrays

PHP allows the direct dereferencing of strings and arrays.

This was added in PHP 5.5. There is no need anymore for an intermediate variable between a string and array (or any expression generating such value) and accessing an index.

```
<?php
$x = array(4,5,6);
$y = $x[2] ; // is 6

May be replaced by
$y = array(4,5,6)[2];
$y = [4,5,6][2];
?>
```

Short name	Structures/DereferencingAS
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*
Php Version	With PHP 5.3 and older
Severity	Major
Time To Fix	Quick (30 mins)

9.159 Detect Current Class

Detecting the current class should be done with *self::class* or *static::class* operator.

__CLASS__ may be replaced by `self\:\:class`. get_called_class() may be replaced by `static\:\:class`.

__CLASS__ and get_called_class() are set to be deprecated in PHP 7.4.

```
<?php

class X {
    function foo() {
        echo __CLASS__.\n;          // X
        echo self::class.\n;        // X

        echo get_called_class().\n;  // Y
        echo static::class.\n;       // Y
    }
}

class Y extends X {}

$y = new Y();
$y->foo();

?>
```

See also PHP RFC: Deprecations for PHP 7.4.

9.159.1 Suggestions

- Use the self::class operator to detect the current class name, instead of __CLASS__ and get_class().
- Use the static::class operator to detect the current called class name, instead of get_called_class().

Short name	Php/DetectCurrentClass
Rulesets	*Suggestions*, *CompatibilityPHP74*
Php Version	With PHP 8.0 and older
Precision	Very high

9.160 Different Argument Counts

Two methods with the same name shall have the same number of compulsory argument. PHP accepts different number of arguments between two methods, if the extra arguments have default values. Basically, they shall be called interchangeably with the same number of arguments.

The number of compulsory arguments is often mistaken for the same number of arguments. When this is the case, it leads to confusion between the two signatures. It will also create more difficulties when refactoring the signature.

While this code is legit, it is recommended to check if the two signatures could be synchronized, and reduce future surprises.

```
<?php

class x {
    function foo($a ) {}
}

class y extends x {
    // This method is compatible with the above, its signature is different
    function foo($a, $b = 1) {}
}

?>
```

9.160.1 Suggestions

- Extract the extra arguments into other methods
- Remove the extra arguments
- Add the extra arguments to all the signatures

Short name	Classes/DifferentArgumentCounts
Rulesets	*Analyze*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.161 Direct Call To __clone()

Direct call to magic method __clone() was forbidden. It is allowed since PHP 7.0.

From the RFC : `Doing calls like $obj->`__clone(<https://www.php.net/manual/en/language.oop5.magic.php>`_ ) is now allowed. This was the only magic method that had a compile-time check preventing some calls to it, which doesn't make sense. If we allow all other magic methods to be called, there's no reason to forbid this one.`

```
<?php

    class Foo {
        function __clone() {}
    }

    $a = new Foo;
    $a->__clone();
?>
```

See also Directly calling '__clone is allowed <https://wiki.php.net/rfc/abstract_syntax_tree#directly_calling_clone_is_allowed>'_.

9.161.1 Suggestions

- Use the clone operator to call the __clone magic method

Short name	Php/DirectCallToClone
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and more recent
Severity	Critical
Time To Fix	Slow (1 hour)

9.162 Direct Injection

The following code act directly upon PHP incoming variables like `$_GET` and `$_POST`. This makes those snippets very unsafe.

```
<?php

// Direct injection
echo Hello.$_GET['user']., welcome.;

// less direct injection
foo($_GET['user']);
function foo($user) {
    echo Hello.$user., welcome.;
}

?>
```

See also Cross-Site Scripting (XSS)

9.162.1 Suggestions

- Validate input : make sure the incoming data are what you expect from them.
- Escape output : prepare outgoing data for the next system to use.

Short name	Security/DirectInjection
Rulesets	*Security*
Severity	Major
Time To Fix	Quick (30 mins)

9.163 Directly Use File

Some PHP functions have a close cousin that work directly on files : use them. This is faster and less code to write.

- md5() => md5_file()
- highlight_string() => highlight_file(), show_source()
- parsekit_compile_string() => parsekit_compile_file()
- parse_ini_string() => parse_ini_file()

- sha1() => sha1_file()
- simplexml_load_string() => simplexml_load_file()
- yaml_parse() => yaml_parse_file()
- hash() => hash_file()
- hash_hmac() => hash_mac_file()
- hash_update() => hash_update_file()
- recode() => recode_file()
- recode_string() => recode_file()

```
<?php

// Good way
$file_hash = hash_file('sha512', 'example.txt');

// Slow way
$file_hash = hash('sha512', file_get_contents('example.txt'));

?>
```

See also hash_file.

9.163.1 Suggestions

- Use the _file() version of those functions

Short name	Structures/DirectlyUseFile
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Instant (5 mins)

9.164 Disconnected Classes

One class is extending the other, but they do not use any features from one another. Basically, those two classes are using extends, but they are completely independent and may be separated.

When using the 'extends' keyword, the newly created classes are now acting together and making one. This should be visible in calls from one class to the other, or simply by property usage : they can't live without each other.

On the other hand, two completely independent classes that are merged, although they should be kept separated.

```
<?php

class A {
    private $pa = 1;

    function fooA() {
        $this->pa = 2;
    }
}
```

(continues on next page)

(continued from previous page)

```
// class B and Class A are totally independent
class B extends A {
    private $pb = 1;

    function fooB() {
        $this->pb = 2;
    }
}


// class C makes use of class A : it is dependent on the parent class
class C extends A {
    private $pc = 1;

    function fooB() {
        $this->pc = 2 + $this->fooA();
    }
}
?>
```

9.164.1 Suggestions

- Remove the extension
- Make actual usage of the classes, at least from one of them

Short name	Classes/DisconnectedClasses
Rulesets	*ClassReview*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*WordPress*

9.165 Do In Base

Use SQL expression to compute aggregates.

```
<?php

// Efficient way
$res = $db->query('SELECT sum(e) AS sumE FROM table WHERE condition');

// The sum is already done
$row = $res->fetchArray();
$c += $row['sumE'];

// Slow way
$res = $db->query('SELECT e FROM table WHERE condition');

// This aggregates the column e in a slow way
while($row = $res->fetchArray()) {
    $c += $row['e'];
```

(continues on next page)

(continued from previous page)

```
}

?>
```

9.165.1 Suggestions

- Rework the query to move the calculations in the database

Short name	Performances/DoInBase
Rulesets	*Performances*
Severity	Major
Time To Fix	Quick (30 mins)

9.166 Don't Be Too Manual

Adapt the examples from the PHP manual to your code. Don't reuse directly the same names in your code : be more specific about what to expect in those variables.

```
<?php

// Search for phone numbers in a text
preg_match_all('/((\d{3})-(\d{3})-(\d{4}))/', $string, $phoneNumber);

// Search for phone numbers in a text
preg_match_all('/(\d{3})-(\d{3})-(\d{4})/', $string, $matches);

?>
```

9.166.1 Suggestions

- Use precise name with your variables

Short name	Structures/DontBeTooManual
Rulesets	*Coding Conventions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.167 Don't Change Incomings

PHP hands over a lot of information using special variables like $_GET, $_POST, etc… Modifying those variables and those values inside variables means that the original content is lost, while it will still look like raw data, and, as such, will be untrustworthy.

```
<?php

// filtering and keeping the incoming value.
```

(continues on next page)

(continued from previous page)

```
$_DATA'id'] = (int) $_GET['id'];

// filtering and changing the incoming value.
$_GET['id'] = strtolower($_GET['id']);

?>
```

It is recommended to put the modified values in another variable, and keep the original one intact.

Short name	Structures/NoChangeIncomingVariables
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)

9.168 Don't Echo Error

It is recommended to avoid displaying error messages directly to the browser.

PHP's uses the `display_errors` directive to control display of errors to the browser. This must be kept to `off` when in production.

```
<?php

// Inside a 'or' test
mysql_connect('localhost', $user, $pass) or die(mysql_error());

// Inside a if test
$result = pg_query( $db, $query );
if( !$result )
{
     echo Erreur SQL: . pg_error();
     exit;
}

// Changing PHP configuration
ini_set('display_errors', 1);
// This is also a security error : 'false' means actually true.
ini_set('display_errors', 'false');

?>
```

Error messages should be logged, but not displayed.

See also Error reporting and List of php.ini directives.

9.168.1 Suggestions

- Remove any echo, print, printf() call built with error messages from an exception, or external source.

Short name	Security/DontEchoError
Rulesets	*Analyze*, *Security*, *CI-checks*
Severity	Critical
Time To Fix	Instant (5 mins)
Examples	*ChurchCRM*, *Phpdocumentor*

9.169 Don't Loop On Yield

Use `yield from`, instead of looping on a generator with `yield`.

`yield from` delegate the yielding to another generator, and keep calling that generator until it is finished. It also works with implicit generator datastructure, like arrays.

```
<?php

function generator() {
    for($i = 0; $i < 10; ++$i) {
        yield $i;
    }
}

function delegatingGenerator() {
    yield from generator();
}

// Too much code here
function generator2() {
    foreach(generator() as $g) {
        yield $g;
    }
}

?>
```

There is a performance gain when delegating, over looping manually on the generator. You may even consider writing the loop to store all values in an array, then `yield from` the array.

See also Generator delegation via yield from.

9.169.1 Suggestions

- Use *yield from* instead of the whole foreach() loop

Short name	Structures/DontLoopOnYield
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Dolibarr*, *Tikiwiki*

9.170 Don't Pollute Global Space

Avoid creating definitions in the global name space.

The global namespace is the default namespace, where all functions, classes, constants, traits and interfaces live. The global namespace is also known as the root namespace.

In particular, PHP native classes usually live in that namespace. By creating functions in that namespace, the code may encounter naming conflict, when the PHP group decides to use a name that the code also uses. This already happened in PHP version 5.1.1, where a `Date` native class was introduced, and had to be disabled in the following minor version.

Nowadays, conflicts appear between components, which claim the same name.

See also Using namespaces: fallback to global function/constant.

9.170.1 Suggestions

- Create a namespace for your code, and store your definition there.

Short name	Php/DontPolluteGlobalSpace
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.171 Don't Read And Write In One Expression

Avoid giving value and using it at the same time, in one expression. This is an undefined behavior of PHP, and may change without warning.

One of those changes happens between PHP 7.2 and 7.3 :

```
<?php

$arr = [1];
$ref =& $arr[0];
var_dump($arr[0] + ($arr[0] = 2));
// PHP 7.2: int(4)
// PHP 7.3: int(3)

?>
```

See also UPGRADING 7.3.

9.171.1 Suggestions

- Split the expression in two separate expressions

Short name	Structures/DontReadAndWriteInOneExpression
Rulesets	*Analyze*, *CompatibilityPHP73*, *CompatibilityPHP74*
Severity	Critical
Time To Fix	Quick (30 mins)

9.172 Don't Send $this In Constructor

Don't use `$this` as an argument while in the __construct(). Until the constructor is finished, the object is not finished, and may be in an unstable state. Providing it to another code may lead to error.

This is true when the receiving structure puts the incoming object immediately to work, and don't store it for later use.

```php
<?php

// $this is only provided when Foo is constructed
class Foo {
    private $bar = null;
    private $data = array();

    static public function build($data) {
        $foo = new Foo($data);
        // Can't build in one call. Must make it separate.
        $foo->finalize();
    }

    private function __construct($data) {
        // $this is provided too early
        $this->data = $data;
    }

    function finalize() {
        $this->bar = new Bar($this);
    }
}

// $this is provided too early, leading to error in Bar
class Foo2 extends Foo {
    private $bar = null;
    private $data = array();

    function __construct($data) {
        // $this is provided too early
        $this->bar = new Bar($this);
        $this->data = $data;
    }
}

class Bar {
    function __construct(Foo $foo) {
        // the cache is now initialized with a wrong
        $this->cache = $foo->getIt();
    }
}

?>
```

See also Don't pass this out of a constructor.

9.172.1 Suggestions

- Finish the constructor first, then call an external object.

- Sending $this should be made accessible in a separate method, so external objects may call it.
- Sending the current may be the responsibility of the method creating the object.

Short name	Classes/DontSendThisInConstructor
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*Woocommerce*, *Contao*

9.173 Don't Unset Properties

Avoid unsetting properties. They would go undefined, and raise more warnings.

When getting rid of a property, assign it to null. This keeps the property in the object, yet allows existence check without errors.

```
<?php

class Foo {
    public $a = 1;
}

$a = new Foo();

var_dump((array) $a) ;
// la propriété est reportée, et null
// ['a' => null]

unset($a->a);

var_dump((array) $a) ;
//Empty []

// Check if a property exists
var_dump($a->b === null);

// Same result as above, but with a warning
var_dump($a->c === null);

?>
```

This analysis works on properties and static properties. It also reports magic properties being unset.

Thanks for Benoit Burnichon for the original idea.

9.173.1 Suggestions

- Never unset properties : set it to null or its default value instead
- Make the property an array, and set/unset its index

Short name	Classes/DontUnsetProperties
Rulesets	*Analyze*, *Top10*, *php-cs-fixable*, *CI-checks*
Severity	Major
Time To Fix	Slow (1 hour)
Examples	*Vanilla*, *Typo3*

9.174 Dont Change The Blind Var

When using a foreach(), the blind variables hold a copy of the original value. It is confusing to modify them, as it seems that the original value may be changed.

When actually changing the original value, use the reference in the foreach definition to make it obvious, and save the final reassignation.

When the value has to be prepared before usage, then save the filtered value in a separate variable. This makes the clean value obvious, and preserve the original value for a future usage.

```php
<?php

// $bar is duplicated and kept
$foo = [1, 2, 3];
foreach($foo as $bar) {
    // $bar is updated but its original value is kept
    $nextBar = $bar + 1;
    print $bar . ' => ' . ($nextBar) . PHP_EOL;
    foobar($nextBar);
}

// $bar is updated and lost
$foo = [1, 2, 3];
foreach($foo as $bar) {
    // $bar is updated but its final value is lost
    print $bar . ' => ' . (++$bar) . PHP_EOL;
    // Now that $bar is reused, it is easy to confuse its value
    foobar($bar);
}

// $bar is updated and kept
$foo = [1, 2, 3];
foreach($foo as &$bar) {
    // $bar is updated and keept
    print $bar . ' => ' . (++$bar) . PHP_EOL;
    foobar($bar);
}

?>
```

Short name	Structures/DontChangeBlindKey
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.175 Dont Collect Void

When a method returns void, there is no need to collect the result. The collected value will actually be `null`.

```
<?php

function foo() : void {
    // doSomething()
}

// This is useless
$result = foo();

// This is useless. It looks like this is a left over from code refactoring
echo foo();

?>
```

9.175.1 Suggestions

- Move the call to the function to its own expression with a semi-colon.
- Remove assignation of the result of such calls.

Short name	Functions/DontUseVoid
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

9.176 Dont Compare Typed Boolean

There is no need to compare explicitly a function call to a boolean, when the definition has a boolean return typehint.

The analysis checks for equality and identity comparisons. It doesn't check for the not operator usage.

```
<?php

// Sufficient check
if (foo()) {
    doSomething();
}

// Superfluous check
if (foo() === true) {
    doSomething();
}

function foo() : bool {}

?>
```

9.176.1 Suggestions

- Simplify the code and make it short

Short name	Structures/DontCompareTypedBoolean
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.177 Dont Mix ++

++ operators, pre and post, have two distinct behaviors, and should be used separately.

When mixed in a larger expression, they are difficult to read, and may lead to unwanted behaviors.

```
<?php

    // Clear and defined behavior
    $i++;
    $a[$i] = $i;

    // The index is also incremented, as it is used AFTP the incrementation
    // With $i = 2; $a is array(3 => 3)
    $a[$i] = ++$i;

    // $i is actually modified twice
    $i = --$i + 1;
?>
```

See also EXP30-C. Do not depend on the order of evaluation for side effects.

9.177.1 Suggestions

- Extract the increment from the expression, and put it on a separate line.

Short name	Structures/DontMixPlusPlus
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Contao*, *Typo3*

9.178 Double Assignation

This happens when a container (variable, property, array index) is assigned with values twice in a row. One of them is probably a debug instruction, that was forgotten.

```
<?php

// Normal assignation
$a = 1;
```

(continues on next page)

(continued from previous page)

```
// Double assignation
$b = 2;
$b = 3;

?>
```

Short name	Structures/DoubleAssignation
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)

9.179 Double Instructions

Twice the same call in a row. This is worth a check.

```
<?php

// repetition of the same command, with the same effect each time.
$a = array_merge($b, $c);
$a = array_merge($b, $c);

// false positive : commands are identical, but the effect is compounded
$a = array_merge($a, $c);
$a = array_merge($a, $c);

?>
```

9.179.1 Suggestions

- Remove double work
- Avoid repetition by using loops, variadic or quantifiers *(dirname($path, 2))*

Short name	Structures/DoubleInstruction
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Instant (5 mins)

9.180 Double Object Assignation

Make sure that assigning the same object to two variables is the intended purpose.

```
<?php

// $x and $y are the same object, as they both hold a reference to the same object.
// This means that changing $x, will also change $y.
$x = $y = new Z();
```

(continues on next page)

(continued from previous page)

```

// $a and $b are distinct values, by default
$a = $b = 1;

?>
```

9.180.1 Suggestions

-

Short name	Structures/DoubleObjectAssignation
Rulesets	*Analyze*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.181 Double array_flip()

Avoid double array_flip() to gain speed. While array_flip() alone is usually useful, a double call to array_flip() is made to make values and keys unique.

```
<?php

// without array_flip
function foo($array, $value) {
    $key = array_search($array, $value);

    if ($key !== false) {
        unset($array[$key]);
    }

    return $array;
}

// double array_flip
// array_flip() usage means that $array's values are all unique
function foo($array, $value) {
    $flipped = array_flip($value);
    unset($flipped[$value]);
    return array_flip($flipped);
}

?>
```

9.181.1 Suggestions

- use array_unique() or array_count_values
- use array_flip() once, and let PHP garbage collect it later
- Keep the original values in a separate variable

Short name	Performances/DoubleArrayFlip
Rulesets	*Performances*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*NextCloud*

9.182 Drop Else After Return

Avoid else clause when the then clause returns, but not the else. And vice-versa.

This way, the else block disappears, and is now the main sequence of the function.

This is also true if else has a return, and then not. When doing so, don’t forget to reverse the condition.

```php
<?php

// drop the else
if ($a) {
    return $a;
} else {
    doSomething();
}

// drop the then
if ($b) {
    doSomething();
} else {
    return $a;
}

// return in else and then
if ($a3) {
    return $a;
} else {
    $b = doSomething();
    return $b;
}

?>
```

9.182.1 Suggestions

- Remove the else clause and move its code to the main part of the method

Short name	Structures/DropElseAfterReturn
Rulesets	*Analyze*, *Suggestions*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.183 Drop Substr Last Arg

Substr() works till the end of the string when the last argument is omitted. There is no need to calculate string size to make this work.

```
<?php

$string = 'abcdef';

// Extract the end of the string
$cde = substr($string, 2);

// Too much work
$cde = substr($string, 2, strlen($string));

?>
```

See also substr.

9.183.1 Suggestions

- Use negative length
- Omit the last argument to get the string till its end

Short name	Structures/SubstrLastArg
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*SuiteCrm*, *Tine20*

9.184 Duplicate Literal

Report literals that are repeated across the code. The minimum replication is 5, and is configurable with `maxDuplicate`.

Repeated literals should be considered a prime candidate for constants.

Integer, reals and strings are considered here. Boolean, Null and Arrays are omitted. 0, 1, 2, 10 and the empty string are all omitted, as too common.

```
<?php
    // array index are omitted
    $x[3] = 'b';

    // constanst are omitted
    const X = 11;
    define('Y', 'string')

    // 0, 1, 2, 10 are omitted
    $x = 0;

?>
```

9.184.1 Suggestions

- Create a constant and use it in place of the literal
- Create a class constant and use it in place of the literal

Name	Default	Type	Description
minDuplicate	15	integer	Minimal number of duplication before the literal is reported.

Short name	Type/DuplicateLiteral
Rulesets	*Semantics*
Severity	Minor
Time To Fix	Quick (30 mins)

9.185 Dynamic Library Loading

Loading a variable dynamically requires a lot of care in the preparation of the library name.

In case of injection in the variable, the dynamic loading of a library gives a lot of power to an intruder.

```
<?php

    // dynamically loading a library
     dl($library. PHP_SHLIB_SUFFIX);

    // dynamically loading ext/vips
     dl('vips.' . PHP_SHLIB_SUFFIX);

    // static loading ext/vips (unix only)
     dl('vips.so');

?>
```

See also dl.

9.185.1 Suggestions

- Use a switch structure, to make the dl() calls static.
- Avoid using dl() and make the needed extension always available in PHP binary.

Short name	Security/DynamicDl
Rulesets	*Security*
Severity	Major
Time To Fix	Slow (1 hour)

9.186 Echo Or Print

Echo and print have the same functional use. <?= and printf() are also considered in this analysis.

There seems to be a choice that is not enforced : one form is dominant, (> 90%) while the others are rare.

The analyzed code has less than 10% of one of the three : for consistency reasons, it is recommended to make them all the same.

It happens that print, echo or <?= are used depending on coding style and files. One file may be consistently using print, while the others are all using echo.

```
<?php

echo 'a';
echo 'b';
echo 'c';
echo 'd';
echo 'e';
echo 'f';
echo 'g';
echo 'h';
echo 'i';
echo 'j';
echo 'k';

// This should probably be written 'echo';
print 'l';

?>
```

Short name	Structures/EchoPrintConsistance
Rulesets	*Coding Conventions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.187 Echo With Concat

Optimize your `echo`'s by not concatenating at `echo` time, but serving all argument separated. This will save PHP a memory copy.

If values, literals and variables, are small enough, this won't have visible impact. Otherwise, this is less work and less memory waste.

```
<?php
  echo $a, ' b ', $c;
?>
```

instead of

```
<?php
  echo  $a . ' b ' . $c;
  echo $a b $c;
?>
```

9.187.1 Suggestions

- Turn the concatenation into a list of argument, by replacing the dots by commas.

Short name	Structures/EchoWithConcat
Rulesets	*Performances*, *Analyze*, *Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)
ClearPHP	no-unnecessary-string-concatenation
Examples	*Phpdocumentor*, *TeamPass*

9.188 Ellipsis Usage

Usage of the ellipsis keyword. The keyword is three dots : `...` . It is also named variadic or splat operator.

It may be in function definitions, either in functioncalls.

`...` allows for packing or unpacking arguments into an array.

```
<?php

$args = [1, 2, 3];
foo(...$args);
// Identical to foo(1,2,3);

function bar(...$a) {
    // Identical to : $a = func_get_args();
}
?>
```

See also PHP RFC: Syntax for variadic functions, PHP 5.6 and the Splat Operator, and Variable-length argument lists.

Short name	Php/EllipsisUsage
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Php Version	With PHP 5.6 and more recent
Severity	Major
Time To Fix	Slow (1 hour)

9.189 Else If Versus Elseif

Always use elseif instead of else and if.

“The keyword elseif SHOULD be used instead of else if so that all control keywords look like single words”. Quoted from the PHP-FIG documentation

```
<?php

// Using elseif
if ($a == 1) { doSomething(); }
elseif ($a == 2) { doSomethingElseIf(); }
else { doSomethingElse(); }

// Using else if
if ($a == 1) { doSomething(); }
else if ($a == 2) { doSomethingElseIf(); }
else { doSomethingElse(); }
```

(continues on next page)

(continued from previous page)

```

// Using else if, no {}
if ($a == 1)  doSomething();
else if ($a == 2) doSomethingElseIf();
else  doSomethingElse();

?>
```

See also elseif/else if.

9.189.1 Suggestions

- Merge else and if into elseif
- Turn the else expression into a block, and have more than the second if in this block
- Turn the if / else if / else into a switch structure

Short name	Structures/ElseIfElseif
Rulesets	*Analyze*, *php-cs-fixable*, *Rector*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*TeamPass*, *Phpdocumentor*

9.190 Empty Blocks

Full empty block, part of a control structures.

It is recommended to remove those blocks, so as to reduce confusion in the code.

```
<?php

foreach($foo as $bar) ; // This block seems erroneous
    $foobar++;

if ($a === $b) {
    doSomething();
} else {
    // Empty block. Remove this
}

// Blocks containing only empty expressions are also detected
for($i = 0; $i < 10; $i++) {
    ;
}

// Although namespaces are not control structures, they are reported here
namespace A;
namespace B;

?>
```

9.190.1 Suggestions

- Fill the block with a command
- Fill the block with a comment that explain the situation
- Remove the block and its commanding operator

Short name	Structures/EmptyBlocks
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Cleverstyle*, *PhpIPAM*

9.191 Empty Classes

Classes that do no define anything at all. This is probably dead code.

Classes that are directly derived from an exception are omitted.

```php
<?php

//Empty class
class foo extends bar {}

//Not an empty class
class foo2 extends bar {
    const FOO = 2;
}

//Not an empty class, as derived from Exception
class barException extends \Exception {}

?>
```

9.191.1 Suggestions

- Remove an empty class :it is probably dead code.
- Add some code to the class to make it concrete.

Short name	Classes/EmptyClass
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*WordPress*

9.192 Empty Function

Function or method whose body is empty.

Such functions or methods are rarely useful. As a bare minimum, the function should return some useful value, even if constant.

A method is considered empty when it contains nothing, or contains expressions without impact.

```php
<?php

// classic empty function
function emptyFunction() {}

class bar {
    // classic empty method
    function emptyMethod() {}

    // classic empty function
    function emptyMethodWithParent() {}
}

class barbar extends bar {
    // NOT an empty method : it overwrites the parent method
    function emptyMethodWithParent() {}
}

?>
```

Methods which overwrite another methods are omitted. Methods which are the concrete version of an abstract method are considered.

9.192.1 Suggestions

- Fill the function with actual code
- Remove any usage of the function, then remove the function

Short name	Functions/EmptyFunction
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Contao*

9.193 Empty Instructions

Empty instructions are part of the code that have no instructions.

This may be trailing semi-colon or empty blocks for if-then structures.

Comments that explains the reason of the situation are not taken into account.

```php
<?php
    $condition = 3;;;;
    if ($condition) { }
?>
```

9.193.1 Suggestions

- Remove the empty lines
- Fill the empty lines

Short name	Structures/EmptyLines
Rulesets	*Dead code*, *Analyze*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Zurmo*, *ThinkPHP*

9.194 Empty Interfaces

Empty interfaces are a code smell. Interfaces should contains at least a method or a constant, and not be totally empty.

```
<?php

// an empty interface
interface empty {}

// an normal interface
interface normal {
    public function i() ;
}

// a constants interface
interface constantsOnly {
    const FOO = 1;
}

?>
```

See also Empty interfaces are bad practice and Blog : Are empty interfaces code smell?.

9.194.1 Suggestions

- Remove the interface
- Add some methods or constants to the interface

Short name	Interfaces/EmptyInterface
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Instant (5 mins)

9.195 Empty List

Empty list() are not allowed anymore in PHP 7. There must be at least one variable in the list call.

```
<?php

//Not accepted since PHP 7.0
list() = array(1,2,3);

//Still valid PHP code
list(,$x) = array(1,2,3);

?>
```

9.195.1 Suggestions

- Remove empty list() calls

Short name	Php/EmptyList
Rulesets	*Analyze*, *CompatibilityPHP70*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Quick (30 mins)

9.196 Empty Namespace

Declaring a namespace in the code and not using it for structure declarations or global instructions is useless.

Using simple style :

```
<?php

namespace Y;

class foo {}


namespace X;
// This is useless

?>
```

Using bracket-style syntax :

```
<?php

namespace X {
    // This is useless
}

namespace Y {

    class foo {}

}

?>
```

9.196.1 Suggestions

- Remove the namespace

Short name	Namespaces/EmptyNamespace
Rulesets	*Analyze*, *Dead code*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
ClearPHP	no-empty-namespace

9.197 Empty Slots In Arrays

PHP tolerates the last element of an array to be empty.

```
<?php
    $a = array( 1, 2, 3, );
    $b =       [ 4, 5, ];
?>
```

Short name	Arrays/EmptySlots
Rulesets	*Coding Conventions*
Severity	Minor
Time To Fix	Instant (5 mins)

9.198 Empty Traits

List of all empty trait defined in the code.

```
<?php

// empty trait
trait t { }

// Another empty trait
trait t2 {
    use t;
}

?>
```

Such traits may be reserved for future use. They may also be forgotten, and dead code.

9.198.1 Suggestions

- Add some code to the trait
- Remove the trait

Short name	Traits/EmptyTrait
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Instant (5 mins)

9.199 Empty Try Catch

The code does try, then catch errors but do no act upon the error.

```
<?php

try {
    doSomething();
} catch (Throwable $e) {
    // ignore this
}

?>
```

At worst, the error should be logged, so as to measure the actual usage of the catch expression.

`catch( Exception $e)` (PHP 5) or `` catch(`Throwable <https://www.php.net/manual/en/class.throwable.php>`_ $e) `` with empty catch block should be banned. They ignore any error and proceed as if nothing happened. At worst, the event should be logged for future analysis.

See also Empty Catch Clause.

9.199.1 Suggestions

- Add some logging in the catch
- Add a comment to mention why the catch is empty
- Change the exception, chain it and throw again

Short name	Structures/EmptyTryCatch
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*LiveZilla*, *Mautic*

9.200 Empty With Expression

empty() doesn’t accept expressions until PHP 5.5. Until then, it is necessary to store the result of the expression in a variable and then, test it with empty().

```
<?php

// PHP 5.5+ empty() usage
if (empty(strtolower($b . $c))) {
    doSomethingWithoutA();
```

(continues on next page)

(continued from previous page)

```
}

// Compatible empty() usage
$a = strtolower($b . $c);
if (empty($a)) {
    doSomethingWithoutA();
}

?>
```

See also empty.

9.200.1 Suggestions

- Use the compatible syntax, and store the result in a local variable before testing it with empty

Short name	Structures/EmptyWithExpression
Rulesets	*Suggestions*
Php Version	With PHP 5.5 and more recent
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*HuMo-Gen*

9.201 Encoded Simple Letters

Some simple letters are written in escape sequence.

Usually, escape sequences are made to encode unusual characters. Using escape sequences for simple characters, like letters or numbers is suspicious.

This analysis also detects Unicode codepoint with superfluous leading zeros.

```
<?php

// This escape sequence makes eval hard to spot
$a = ev1l;
$a('php_info();');

// With a PHP 7.0 unicode code point sequence
$a = ev\u{000041}l;
$a('php_info();');

// With a PHP 5.0+ hexadecimal sequence
$a = ev\x41l;
$a('php_info();');

?>
```

9.201.1 Suggestions

- Make all simple letter appear clearly

- Add comments about why this code is encoded

Short name	Security/EncodedLetters
Rulesets	*Security*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Zurmo*

9.202 Eval() Usage

Using eval() is evil.

Using eval() is bad for performances (compilation time), for caches (it won't be compiled), and for security (if it includes external data).

```
<?php
    // Avoid using incoming data to build the eval() expression : any filtering error␣
↪leads to PHP injection
    $mathExpression = $_GET['mathExpression'];
    $mathExpression = preg_replace('#[^0-9+\-*/\(/)]#is', '', $mathExpression); //␣
↪expecting 1+2
    $literalCode = '$a = '.$mathExpression.';';
    eval($literalCode);
    echo $a;

    // If the code code given to eval() is known at compile time, it is best to put␣
↪it inline
    $literalCode = 'phpinfo();';
    eval($literalCode);

?>
```

Most of the time, it is possible to replace the code by some standard PHP, like variable variable for accessing a variable for which you have the name. At worse, including a pregenerated file is faster and cacheable.

There are several situations where eval() is actually the only solution :

For PHP 7.0 and later, it is important to put eval() in a try..catch expression.

See also eval and The Land Where PHP Uses 'eval() <https://www.exakat.io/land-where-php-uses-eval/>'_.

9.202.1 Suggestions

- Use a dynamic feature of PHP to replace the dynamic code
- Store the code on the disk, and use include
- Replace create_function() with a closure!

Short name	Structures/EvalUsage
Rulesets	*Analyze*, *Performances*, *Security*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	no-eval
Examples	*XOOPS*, *Mautic*

9.203 Exceeding Typehint

The typehint is not fully used in the method. Some of the defined methods in the typehint are unused. A tighter typehint could be used, to avoid method pollution.

```
<?php

interface i {
    function i1();
    function i2();
}

interface j {
    function j1();
    function j2();
}

function foo(i $a, j $b) {
    // the i typehint is totally used
    $a->i1();
    $a->i2();

    // the i typehint is not totally used : j2() is not used.
    $b->j1();
}

?>
```

Tight typehint prevents the argument from doing too much. They also require more maintenance : creation of dedicated interfaces, method management to keep all typehint tight.

See also *Insufficient Typehint*.

9.203.1 Suggestions

- Keep the typehint tight, do not inject more than needed.

Short name	Functions/ExceedingTypehint
Rulesets	*ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.204 Exception Order

When catching exception, the most specialized exceptions must be in the early catch, and the most general exceptions must be in the later catch. Otherwise, the general catches intercept the exception, and the more specialized will not be read.

```
<?php

class A extends \Exception {}
class B extends A {}
```

(continues on next page)

(continued from previous page)

```

try {
    throw new A();
}
catch(A $a1) { }
catch(B $b2 ) {
    // Never reached, as previous Catch is catching the early worm
}

?>
```

Short name	Exceptions/AlreadyCaught
Rulesets	*Dead code*
Examples	*Woocommerce*

9.205 Exit() Usage

Using exit or die() <https://www.php.net/‘die>‘_ in the code makes the code untestable (it will break unit tests). Moreover, if there is no reason or string to display, it may take a long time to spot where the application is stuck.

```
<?php

// Throw an exception, that may be caught somewhere
throw new \Exception('error');

// Dying with error message.
die('error');

function foo() {
    //exiting the function but not dying
    if (somethingWrong()) {
        return true;
    }
}
?>
```

Try exiting the function/class with return, or throw exception that may be caught later in the code.

9.205.1 Suggestions

- Avoid exit and die. Let the script finish.
- Throw an exception and let it be handled before finishing

Short name	Structures/ExitUsage
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	no-exit
Examples	*Traq*, *ThinkPHP*

9.206 Exponent Usage

Usage of the ** operator or **=, to make exponents.

```
<?php

$eight = 2 ** 3;

$sixteen = 4;
$sixteen \*\*\= 2;

?>
```

See also Arithmetic Operators.

Short name	Php/ExponentUsage
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Php Version	With PHP 5.6 and more recent
Severity	Major
Time To Fix	Instant (5 mins)

9.207 Failed Substr Comparison

The extracted string must be of the size of the compared string.

This is also true for negative lengths.

```
<?php

// Possible comparison
if (substr($a, 0, 3) === 'abc') { }
if (substr($b, 4, 3) === 'abc') { }

// Always failing
if (substr($a, 0, 3) === 'ab') { }
if (substr($a, 3, -3) === 'ab') { }

// Omitted in this analysis
if (substr($a, 0, 3) !== 'ab') { }

?>
```

9.207.1 Suggestions

- Fix the string
- Fix the length of the string
- Put the string in a constant, and use strlen() or mb_strlen()

Short name	Structures/FailingSubstrComparison
Rulesets	*Analyze*, *Top10*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)
Examples	*Zurmo*, *MediaWiki*

9.208 Fetch One Row Format

When reading results with ext/Sqlite3, it is recommended to explicitly request SQLITE3_NUM or SQLITE3_ASSOC, while avoiding the default value and SQLITE3_BOTH.

```
<?php

$res = $database->query($query);

// Fastest version, but less readable
$row = $res->fetchArray(\SQLITE3_NUM);
// Almost the fastest version, and more readable
$row = $res->fetchArray(\SQLITE3_ASSOC);

// Default version. Quite slow
$row = $res->fetchArray();

// Worse case
$row = $res->fetchArray(\SQLITE3_BOTH);

?>
```

This is a micro-optimisation. The difference may be visible with 200k rows fetches, and measurable with 10k.

9.208.1 Suggestions

- Specify the result format when reading rows from a Sqlite3 database

Short name	Performances/FetchOneRowFormat
Rulesets	*Performances*
Severity	Minor
Time To Fix	Instant (5 mins)

9.209 Filter To add_slashes()

`FILTER_SANITIZE_MAGIC_QUOTES` is deprecated. In PHP 7.4, it should be replaced with addslashes()

According to the migration RDFC : 'Magic quotes were deprecated all the way back in PHP 5.3 and later removed in PHP 5.4. The filter extension implements a sanitization filter that mimics this behavior of magic_quotes by calling addslashes() on the input in question.'

```
<?php

// Deprecated way to filter input
```

(continues on next page)

(continued from previous page)

```
$var = filter_input($input, FILTER_SANITIZE_MAGIC_QUOTES);

// Alternative way to filter input
$var = addslashes($input);

?>
```

addslashes() used to filter data while building SQL queries, to prevent injections. Nowadays, prepared queries are a better option.

See also Deprecations for PHP 7.4.

9.209.1 Suggestions

- Replace `FILTER_SANITIZE_MAGIC_QUOTES` with addslashes()
- Replace `FILTER_SANITIZE_MAGIC_QUOTES` with an adapted escaping system

Short name	Php/FilterToAddSlashes
Rulesets	*CompatibilityPHP74*
Severity	Minor
Time To Fix	Quick (30 mins)

9.210 Final Class Usage

List of all final classes being used.

final may be applied to classes and methods.

```
<?php
class BaseClass {
   public function test() {
       echo 'BaseClass::test() called'.PHP_EOL;
   }

   final public function moreTesting() {
       echo 'BaseClass::moreTesting() called'.PHP_EOL;
   }
}

class ChildClass extends BaseClass {
   public function moreTesting() {
       echo 'ChildClass::moreTesting() called'.PHP_EOL;
   }
}
// Results in Fatal error: Cannot override final method BaseClass::moreTesting()
?>
```

See also Final Keyword.

Short name	Classes/Finalclass
Rulesets	*ClassReview*, *LintButWontExec*

9.211 Final Methods Usage

List of all final methods being used.

final may be applied to classes and methods.

```
<?php
class BaseClass {
   public function test() {
       echo 'BaseClass::test() called'.PHP_EOL;
   }

   final public function moreTesting() {
       echo 'BaseClass::moreTesting() called'.PHP_EOL;
   }
}

class ChildClass extends BaseClass {
   public function moreTesting() {
       echo 'ChildClass::moreTesting() called'.PHP_EOL;
   }
}
// Results in Fatal error: Cannot override final method BaseClass::moreTesting()
?>
```

See also Final Keyword.

Short name	Classes/Finalmethod
Rulesets	*LintButWontExec*, *ClassReview*

9.212 Flexible Heredoc

Flexible syntax for Heredoc.

The new flexible syntax for heredoc and nowdoc enable the closing marker to be indented, and remove the new line requirement after the closing marker.

It was introduced in PHP 7.3.

```
<?php

// PHP 7.3 and newer
foo($a = <<<END

    flexible syntax
    with extra indentation

    END);

// All PHP versions
$a = <<<END

    Normal syntax

END;
```

(continues on next page)

(continued from previous page)

```


?>
```

This syntax is backward incompatible : once adopted in the code, previous versions won't compile it.

See also Heredoc and Flexible Heredoc and Nowdoc Syntaxes.

Short name	Php/FlexibleHeredoc
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.3 and more recent
Severity	Critical
Time To Fix	Instant (5 mins)

9.213 Fn Argument Variable Confusion

Avoid using local variables as arrow function arguments.

When a local variable name is used as an argument's name in an arrow function, the local variable from the original scope is not imported. They are now two distinct variables.

When the local variable is not listed as argument, it is then imported in the arrow function.

```
<?php

function foo() {
    $locale = 1;

    // Actually ignores the argument, and returns the local variable ``$locale``.
    $fn2 = fn ($argument) => $locale;

    // Seems similar to above, but returns the incoming argument
    $fn2 = fn ($locale) => $locale;
}

?>
```

See also Arrow functions.

9.213.1 Suggestions

- Change the name of the local variable
- Change the name of the argument

Short name	Functions/FnArgumentVariableConfusion
Rulesets	*Analyze*, *Semantics*
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)

9.214 For Using Functioncall

It is recommended to avoid functioncall in the for() statement.

```php
<?php

// Fastest way
$nb = count($array);
for($i = 0; $i < $nb; ++$i) {
    doSomething($i);
}

// Same as above, but slow
for($i = 0; $i < count($array); ++$i) {
    doSomething($i);
}

// Same as above, but slow
foreach($portions as &$portion) {
    // here, array_sum() doesn't depends on the $grade. It should be out of the loop
    $portion = $portion / array_sum($portions);
}

$total = array_sum($portion);
foreach($portion as &$portion) {
    $portion = $portion / $total;
}

?>
```

This is true with any kind of functioncall that returns the same value throughout the loop.

Short name	Structures/ForWithFunctioncall
Rulesets	*Performances*, *Top10*
Severity	Minor
Time To Fix	Slow (1 hour)
ClearPHP	no-functioncall-in-loop

9.215 Foreach Don't Change Pointer

foreach loops use their own internal cursor.

A foreach loop won't change the internal pointer of the array, as it works on a copy of the source. Hence, applying array pointer's functions such as current() or next() to the source array won't have the same behavior in PHP 5 than PHP 7.

This only applies when a foreach() by reference is used.

```
<?php

$numbers = range(1, 10);
next($numbers);
foreach($numbers as &$number){
    print $number;
    print current($numbers).\n; // Always
}

?>
```

See also foreach no longer changes the internal array pointer and foreach.

Short name	Php/ForeachDontChangePointer
Rulesets	*CompatibilityPHP70*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.216 Foreach On Object

Foreach on object looks like a typo. This is particularly true when both object and member are variables.

Foreach on an object member is a legit PHP syntax, though it is very rare : blind variables rarely have to be securing in an object to be processed.

```
<?php

// Looks suspicious
foreach($array as $o -> $b) {
    doSomething();
}

// This is the real thing
foreach($array as $o => $b) {
    doSomething();
}

?>
```

Short name	Php/ForeachObject
Rulesets	*Analyze*
Severity	Major
Time To Fix	Instant (5 mins)

9.217 Foreach Reference Is Not Modified

Foreach statement may loop using a reference, especially when the loop has to change values of the array it is looping on.

In the spotted loop, reference are used but never modified. They may be removed.

```
<?php

$letters = range('a', 'z');

// $letter is not used here
foreach($letters as &$letter) {
    $alphabet .= $letter;
}

// $letter is actually used here
foreach($letters as &$letter) {
    $letter = strtoupper($letter);
}

?>
```

9.217.1 Suggestions

- Remove the reference from the foreach
- Actually modify the content of the reference

Short name	Structures/ForeachReferenceIsNotModified
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Dolibarr*, *Vanilla*

9.218 Foreach With list()

Foreach loops have the ability to use list as blind variables. This syntax assign directly array elements to various variables.

PHP 5.5 introduced the usage of list in foreach() loops. Until PHP 7.1, it was not possible to use non-numerical arrays as list() wouldn't support string-indexed arrays.

```
<?php
    // PHP 5.5 and later, with numerically-indexed arrays
    foreach($array as list($a, $b)) {
        // do something
    }


    // PHP 7.1 and later, with arrays
    foreach($array as list('col1' => $a, 'col3' => $b)) { // 'col2 is ignored'
        // do something
    }
?>
```

Previously, it was compulsory to extract() the data from the blind array :

```php
<?php
    foreach($array as $c) {
        list($a, $b) = $c;
        // do something
    }
?>
```

See also The list function & practical uses of array destructuring in PHP.

Short name	Structures/ForeachWithList
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*
Php Version	With PHP 5.5 and more recent
Severity	Minor
Time To Fix	Slow (1 hour)

9.219 Forgotten Interface

The following classes have been found implementing an interface's methods, though it doesn't explicitly implements this interface. This may have been forgotten.

```php
<?php

interface i {
    function i();
}

// i is not implemented and declared
class foo {
    function i() {}
    function j() {}
}

// i is implemented and declared
class foo implements i {
    function i() {}
    function j() {}
}

?>
```

See also could-use-trait.

9.219.1 Suggestions

- Mention interfaces explicitly whenever possible

Short name	Interfaces/CouldUseInterface
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.220 Forgotten Thrown

An exception is instantiated, but not thrown.

```
<?php

class MyException extends \Exception { }

if ($error !== false) {
    // This looks like 'throw' was omitted
    new MyException();
}

?>
```

9.220.1 Suggestions

- Remove the throw expression
- Add the new to the throw expression

Short name	Exceptions/ForgottenThrown
Rulesets	*Analyze*
Severity	Major
Time To Fix	Instant (5 mins)

9.221 Forgotten Visibility

Some classes elements (property, method, constant) are missing their explicit visibility.

By default, it is public. It should at least be mentioned as public, or may be reviewed as protected or private.

Class constants support also visibility since PHP 7.1.

final, static and abstract are not counted as visibility. Only public, private and protected. The PHP 4 var keyword is counted as undefined.

Traits, classes and interfaces are checked.

```
<?php

// Explicit visibility
class X {
    protected sconst NO_VISIBILITY_CONST = 1; // For PHP 7.2 and later

    private $noVisibilityProperty = 2;

    public function Method() {}
}

// Missing visibility
class X {
    const NO_VISIBILITY_CONST = 1; // For PHP 7.2 and later
```

(continues on next page)

(continued from previous page)

```
    var $noVisibilityProperty = 2; // Only with var

    function NoVisibilityForMethod() {}
}

?>
```

See also Visibility and Understanding The Concept Of Visibility In Object Oriented PHP.

9.221.1 Suggestions

- Always add explicit visibility to methods and constants in a class
- Always add explicit visibility to properties in a class, after PHP 7.4

Short name	Classes/NonPpp
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
ClearPHP	always-have-visibility
Examples	*FuelCMS*, *LiveZilla*

9.222 Forgotten Whitespace

Forgotten whitespaces only bring misery to the code.

White spaces have been left at either end of a file : before the PHP opening tag, or after the closing tag.

Usually, such whitespaces are forgotten, and may end up summoning the infamous ‘headers already sent’ error. It is better to remove them.

```
<?php
    // This script has no forgotten whitespace, not at the beginning
    function foo() {}

    // This script has no forgotten whitespace, not at the end
?>
```

See also How to fix Headers already sent error in PHP.

9.222.1 Suggestions

- Remove all whitespaces before and after a script. This doesn’t apply to template, which may need to use those spaces.
- Remove the final tag, to prevent any whitespace to be forgotten at the end of the file. This doesn’t apply to the opening PHP tag, which is always necessary.

Short name	Structures/ForgottenWhiteSpace
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)

9.223 Fossilized Method

A method is fossilized when it is overwritten so often that changing a default value, a return type or an argument type is getting difficult.

This happens when a class is extended. When a method is overwritten once, it may be easy to update the signature in two places. The more methods are overwriting a parent method, the more difficult it is to update it.

This analysis counts the number of times a method is overwritten, and report any method that is ovrewritten more than 6 times. This threshold may be configured.

```
<?php

class x1 {
    // foo1() is never overwritten. It is easy to update.
    function foo1() {}

    // foo7() is overwritten seven times. It is hard to update.
    function foo7() {}
}

// classes x2 to x7, all overwrite foo7();
// Only x2 is presente here.
class x2 extends x1 {
    function foo7() {}
}

?>
```

Name	Default	Type	Description
fossilizationThreshold	6	integer	Minimal number of overwriting methods to consider a method difficult to update.

Short name	Classes/FossilizedMethod
Rulesets	*ClassReview*, *Typechecks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.224 Fully Qualified Constants

Constants defined with their namespace.

When defining constants with define() function, it is possible to include the actual namespace :

```
<?php

define('a\b\c', 1);

?>
```

However, the name should be fully qualified without the initial . Here, abc constant will never be accessible as a namespace constant, though it will be accessible via the constant() function.

Also, the namespace will be absolute, and not a relative namespace of the current one.

9.224.1 Suggestions

- Drop the initial when creating constants with define() : for example, use trim($x, ‘’), which removes anti-slashes before and after.

Short name	Namespaces/ConstantFullyQualified
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)

9.225 Function Subscripting

It is possible to use the result of a methodcall directly as an array, without storing the result in a temporary variable.

This works, given that the method actually returns an array.

This syntax was not possible until PHP 5.4. Until then, it was compulsory to store the result in a variable first. Although this is now superfluous, it has been a standard syntax in PHP, and is still being used.

```
<?php

function foo() {
    return array(1 => 'a', 'b', 'c');
}

echo foo()[1]; // displays 'a';

// Function subscripting, the old way
function foo() {
    return array(1 => 'a', 'b', 'c');
}

$x = foo();
echo $x[1]; // displays 'a';

?>
```

Storing the result in a variable is still useful if the result is actually used more than once.

Short name	Structures/FunctionSubscripting
Rulesets	*CompatibilityPHP53*
Php Version	With PHP 5.4 and more recent
Severity	Minor
Time To Fix	Instant (5 mins)

9.226 Function Subscripting, Old Style

Since PHP 5.4, it is now possible use function results as an array, and access directly its element :

```
<?php

function foo() {
    return array(1 => 'a', 'b', 'c');
}

echo foo()[1]; // displays 'a';

// Function subscripting, the old way
function foo() {
    return array(1 => 'a', 'b', 'c');
}

$x = foo();
echo $x[1]; // displays 'a';

?>
```

9.226.1 Suggestions

- Skip the local variable and directly use the return value from the function

Short name	Structures/FunctionPreSubscripting
Rulesets	*Suggestions*
Php Version	With PHP 5.4 and more recent
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*OpenConf*

9.227 Functions Removed In PHP 5.4

Those functions were removed in PHP 5.4.

```
<?php

// Deprecated as of PHP 5.4.0
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$db_list = mysql_list_dbs($link);

while ($row = mysql_fetch_object($db_list)) {
     echo $row->Database . "\n";
}

?>
```

See also Deprecated features in PHP 5.4.x.

Short name	Php/Php54RemovedFunctions
Rulesets	*CompatibilityPHP54*
Php Version	With PHP 5.4 and older
Severity	Major
Time To Fix	Quick (30 mins)

9.228 Functions Removed In PHP 5.5

Those functions were removed in PHP 5.5.

- php_logo_guid()
- php_egg_logo_guid()
- php_real_logo_guid()
- zend_logo_guid()
- mcrypt_cbc()
- mcrypt_cfb()
- mcrypt_ecb()
- mcrypt_ofb()

```
<?php

echo '<img src="' . $_SERVER['PHP_SELF'] .
     '?=' . php_logo_guid() . '" alt="PHP Logo !" />';

?>
```

See also Deprecated features in PHP 5.5.x.

9.228.1 Suggestions

- Stop using those functions

Short name	Php/Php55RemovedFunctions
Rulesets	*CompatibilityPHP55*
Php Version	With PHP 5.5 and older
Severity	Major
Time To Fix	Quick (30 mins)

9.229 Generator Cannot Return

Generators could not use return and yield at the same time. In PHP 7.0, generator can now use both of them.

```
<?php

// This is not allowed until PHP 7.0
function foo() {
    yield 1;
    return 'b';
}

?>
```

9.229.1 Suggestions

- Remove the return

Short name	Functions/GeneratorCannotReturn
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	7.0+
Severity	Major
Time To Fix	Quick (30 mins)

9.230 Getting Last Element

Getting the last element of an array relies on array_key_last().

array_key_last() was added in PHP 7.3. Before that,

```php
<?php

$array = ['a' => 1, 'b' => 2, 'c' => 3];

// Best solutions, by far
$last = $array[array_key_last($array)];

// Best solutions, just as fast as each other
$last = $array[count($array) - 1];
$last = end($array);

// Bad solutions

// popping, but restoring the value.
$last = array_pop($array);
$array[] = $last;

// array_unshift would be even worse

// reversing array
$last = array_reverse($array)[0];

// slicing the array
$last = array_slice($array, -1)[0]',
$last = current(array_slice($array, -1));
);

?>
```

9.230.1 Suggestions

- Use PHP native function : array_key_last(), when using PHP 7.4 and later
- Use PHP native function : array_pop()
- Organise the code to put the last element in the first position (array_unshift() instead of append operator [])

Short name	Arrays/GettingLastElement
Rulesets	*Performances*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Thelia*

9.231 Global Inside Loop

The global keyword must be used out of loops. Otherwise, it is evaluated each loop, slowing the whole process.

```
<?php

// Here, global is used once
global $total;
foreach($a as $b) {
    $total += $b;
}

// Global is called each time : this is slow.
foreach($a as $b) {
    global $total;
    $total += $b;
}
?>
```

9.231.1 Suggestions

- Move the global keyword outside the loop

Short name	Structures/GlobalOutsideLoop
Rulesets	*Performances*

9.232 Global Usage

List usage of globals variables, with global keywords or direct access to $GLOBALS.

```
<?php
$a = 1; /* global scope */

function test()
{
    echo $a; /* reference to local scope variable */
}

test();

?>
```

It is recommended to avoid using global variables, at it makes it very difficult to track changes in values across the whole application.

See also Variable scope.

Short name	Structures/GlobalUsage
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
ClearPHP	no-global

9.233 Group Use Declaration

The group use declaration is used in the code.

```
<?php

// Adapted from the RFC documentation
// Pre PHP 7 code
use some\name_space\ClassA;
use some\name_space\ClassB;
use some\name_space\ClassC as C;

use function some\name_space\fn_a;
use function some\name_space\fn_b;
use function some\name_space\fn_c;

use const some\name_space\ConstA;
use const some\name_space\ConstB;
use const some\name_space\ConstC;

// PHP 7+ code
use some\name_space\{ClassA, ClassB, ClassC as C};
use function some\name_space\{fn_a, fn_b, fn_c};
use const some\name_space\{ConstA, ConstB, ConstC};

?>
```

See also Group Use Declaration RFC and Using namespaces: Aliasing/Importing.

Short name	Php/GroupUseDeclaration
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Severity	Minor
Time To Fix	Instant (5 mins)

9.234 Group Use Trailing Comma

The usage of a final empty slot in array() was allowed with use statements. This works in PHP 7.2 and more recent.

Although this empty instruction is ignored at execution, this allows for clean presentation of code, and short diff when committing in a VCS.

```
<?php
```

(continues on next page)

(continued from previous page)

```
// Valid in PHP 7.2 and more recent.
use a\b\{c,
         d,
         e,
         f,
        };

// This won't compile in 7.1 and older.

?>
```

See also Trailing Commas In List Syntax and Revisit trailing commas in function arguments.

Short name	Php/GroupUseTrailingComma
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.2 and more recent
Severity	Major
Time To Fix	Instant (5 mins)

9.235 Hardcoded Passwords

Hardcoded passwords in the code.

Hardcoding passwords is a bad idea. Not only it make the code difficult to change, but it is an information leak. It is better to hide this kind of information out of the code.

```
<?php

$ftp_server = '300.1.2.3';   // yes, this doesn't exists, it's an example
$conn_id = ftp_connect($ftp_server);

// login with username and password
$login_result = ftp_login($conn_id, 'login', 'password');

?>
```

See also 10 GitHub Security Best Practices and Git How-To: Remove Your Password from a Repository.

9.235.1 Suggestions

- Remove all passwords from the code. Also, check for history if you are using a VCS.

Name	Default	Type	Description
passwordsKeys	password_keys.json	data	List of array index and property names that shall be checked for potential secret key storages.

Short name	Functions/HardcodedPasswords
Rulesets	*Analyze*, *Security*
Severity	Major
Time To Fix	Slow (1 hour)
ClearPHP	no-hardcoded-credential

9.236 Hash Algorithms

There is a long but limited list of hashing algorithm available to PHP. The one found doesn't seem to be existing.

```php
<?php

// This hash has existed in PHP. Check with hash_algos() if it is available on your system.
echo hash('ripmed160', 'The quick brown fox jumped over the lazy dog.');

// This hash doesn't exist
echo hash('ripemd160', 'The quick brown fox jumped over the lazy dog.');

?>
```

See also hash_algos.

9.236.1 Suggestions

- Use a hash algorithm that is available on several PHP versions
- Fix the name of the hash algorithm

Short name	Php/HashAlgos
Rulesets	*Analyze*
Severity	Major
Time To Fix	Slow (1 hour)

9.237 Hash Algorithms Incompatible With PHP 5.3

List of hash algorithms incompatible with PHP 5.3.

```php
<?php

// Compatible only with 5.3 and more recent
echo hash('md2', 'The quick brown fox jumped over the lazy dog.');

// Always compatible
echo hash('ripemd320', 'The quick brown fox jumped over the lazy dog.');

?>
```

See also hash_algos.

Short name	Php/HashAlgos53
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP72*
Severity	Major
Time To Fix	Slow (1 hour)

9.238 Hash Algorithms Incompatible With PHP 5.4/5.5

List of hash algorithms incompatible with PHP 5.4 and 5.5.

```php
<?php

// Compatible only with 5.4 and more recent
echo hash('fnv132', 'The quick brown fox jumped over the lazy dog.');

// Always compatible
echo hash('ripemd320', 'The quick brown fox jumped over the lazy dog.');

?>
```

See also hash_algos.

Short name	Php/HashAlgos54
Rulesets	*CompatibilityPHP54*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP72*
Php Version	With PHP 5.4 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.239 Hash Algorithms Incompatible With PHP 7.1-

List of hash algorithms incompatible with PHP 7.1 and more recent. At the moment of writing, this is compatible up to 7.3.

The hash algorithms were introduced in PHP 7.1.

```php
<?php

// Compatible only with 7.1 and more recent
echo hash('sha512/224', 'The quick brown fox jumped over the lazy dog.');

// Always compatible
echo hash('ripemd320', 'The quick brown fox jumped over the lazy dog.');

?>
```

See also hash_algos.

Short name	Php/HashAlgos71
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.1 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.240 Hash Algorithms Incompatible With PHP 7.4-

List of hash algorithms incompatible with PHP 7.3 and older recent. At the moment of writing, this is compatible up to 7.4s.

The hash algorithms were introduced in PHP 7.4s.

```php
<?php

// Compatible only with 7.1 and more recent
echo hash('crc32cs', 'The quick brown fox jumped over the lazy dog.');

// Always compatible
echo hash('ripemd320', 'The quick brown fox jumped over the lazy dog.');

?>
```

See also hash_algos.

Short name	Php/HashAlgos74
Rulesets	*CompatibilityPHP74*
Php Version	With PHP 7.4 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.241 Hash Will Use Objects

The ext/hash extension used resources, and is being upgraded to use resources.

```php
<?php

// Post 7.2 code
    $hash = hash_init('sha256');
    if (!is_object($hash)) {
        trigger_error('error');
    }
    hash_update($hash, $message);

// Pre-7.2 code
```

(continues on next page)

(continued from previous page)

```
    $hash = hash_init('md5');
    if (!is_resource($hash)) {
        trigger_error('error');
    }
    hash_update($hash, $message);

?>
```

See also Move ext/hash from resources to objects.

Short name	Php/HashUsesObjects
Rulesets	*CompatibilityPHP72*
Severity	Major
Time To Fix	Quick (30 mins)

9.242 Heredoc Delimiter

Heredoc and Nowdoc expressions may use a variety of delimiters.

There seems to be a standard delimiter in the code, and some exceptions : one or several forms are dominant (> 90%), while the others are rare.

The analyzed code has less than 10% of the rare delimiters. For consistency reasons, it is recommended to make them all the same.

Generally, one or two delimiters are used, with generic value. It is recommended to use a humanly readable delimiter : SQL, HTML, XML, GREMLIN, etc. This helps readability in the code.

```
<?php

echo <<<SQL
SELECT * FROM table1;
SQL;

echo <<<SQL
SELECT * FROM table2;
SQL;

echo <<<SQL
SELECT * FROM table3;
SQL;

echo <<<SQL
SELECT * FROM table4;
SQL;

echo <<<SQL
SELECT * FROM table5;
SQL;

echo <<<SQL
SELECT * FROM table11;
SQL;
```

(continues on next page)

(continued from previous page)

```
echo <<<SQL
SELECT * FROM table12;
SQL;

echo <<<SQL
SELECT * FROM table13;
SQL;

// Nowdoc
echo <<<'SQL'
SELECT * FROM table14;
SQL;

echo <<<SQL
SELECT * FROM table15;
SQL;


echo <<<HEREDOC
SELECT * FROM table215;
HEREDOC;

?>
```

Short name	Structures/HeredocDelimiterFavorite
Rulesets	*Coding Conventions*

9.243 Hexadecimal In String

Mark strings that may be confused with hexadecimal.

Until PHP 7.0, PHP recognizes hexadecimal numbers inside strings, and converts them accordingly.

PHP 7.0 and until 7.1, converts the string to 0, silently.

PHP 7.1 and later, emits a ‘A non-numeric value encountered’ warning, and convert the string to 0.

```
<?php
    $a = '0x0030';
    print $a + 1;
    // Print 49

    $c = '0x0030zyc';
    print $c + 1;
    // Print 49

    $b = 'b0x0030';
    print $b + 1;
    // Print 0
?>
```

Short name	Type/HexadecimalString
Rulesets	*CompatibilityPHP70*, *CompatibilityPHP71*
Severity	Major
Time To Fix	Slow (1 hour)

9.244 Hidden Nullable

Argument with default value of null are nullable. Even when the `null` typehint (PHP 8.0), or the `?` operator are not used, setting the default value to null is allowed, and makes the argument nullable.

This doesn't happen with properties : they must be defined with the nullable type to accept a ‘‘null‘‘value as default value.

This doesn't happen with constant, which can't be typehinted.

```php
<?php

// explicit nullable parameter $s
function bar(?string $s = null) {

// implicit nullable parameter $s
function foo(string $s = null) {
    echo $s ?? 'NULL-value';
}

// both display NULL-value
foo();
foo(null);

?>
```

See also Nullable types and Type declaration.

9.244.1 Suggestions

- Change the default value to a compatible literal : for example, `string $s = ''`
- Add the explicit `?` nullable operator, or ‘‘null‘‘with PHP 8.0
- Remove the default value

Short name	Classes/HiddenNullable
Rulesets	*Analyze*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.245 Hidden Use Expression

The use expression for namespaces should always be at the beginning of the namespace block.

It is where everyone expect them, and it is less confusing than having them at various levels.

```
<?php

// This is visible
use A;

class B {}

// This is hidden
use C as D;

class E extends D {
    use traitT; // This is a use for a trait

    function foo() {
        // This is a use for a closure
        return function ($a) use ($b) {}
    }
}

?>
```

9.245.1 Suggestions

- Group all uses together, at the beginning of the namespace or class

Short name	Namespaces/HiddenUse
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Tikiwiki*, *OpenEMR*

9.246 Htmlentities Calls

htmlentities() and htmlspecialchars() are used to prevent injecting special characters in HTML code. As a bare minimum, they take a string and encode it for HTML.

The second argument of the functions is the type of protection. The protection may apply to quotes or not, to HTML 4 or 5, etc. It is highly recommended to set it explicitly.

The third argument of the functions is the encoding of the string. In PHP 5.3, it is `ISO-8859-1`, in 5.4, was `UTF-8`, and in 5.6, it is now default_charset, a `php.ini` configuration that has the default value of `UTF-8`. It is highly recommended to set this argument too, to avoid distortions from the configuration.

```
<?php
$str = 'A quote is <b>bold</b>';

// Outputs, without depending on the php.ini: A &#039;quote&#039; is &lt;b&gt;bold&lt;
↪/b&gt;
echo htmlentities($str, ENT_QUOTES, 'UTF-8');

// Outputs, while depending on the php.ini: A quote is &lt;b&gt;bold&lt;/b&gt;
echo htmlentities($str);
```

(continues on next page)

(continued from previous page)

```
?>
```

Also, note that arguments 2 and 3 are constants and string, respectively, and should be issued from the list of values available in the manual. Other values than those will make PHP use the default values.

See also htmlentities and htmlspecialchars.

9.246.1 Suggestions

- Always use the third argument with htmlentities()

Short name	Structures/Htmlentitiescall
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)

9.247 Identical Conditions

These logical expressions contain members that are identical.

This means those expressions may be simplified.

```php
<?php

// twice $a
if ($a || $b || $c || $a) {   }

// Hiding in parenthesis is bad
if (($a) ^ ($a)) {}

// expressions may be large
if ($a === 1 && 1 === $a) {}

?>
```

9.247.1 Suggestions

- Merge the two structures into one unique test
- Add extra expressions between the two structures
- Nest the structures, to show that different attempts are made

Short name	Structures/IdenticalConditions
Rulesets	*Analyze*, *CI-checks*
Severity	Critical
Time To Fix	Quick (30 mins)
Examples	*WordPress*, *Dolibarr*, *Mautic*

9.248 Identical Consecutive Expression

Identical consecutive expressions are worth being checked.

They may be a copy/paste with unmodified content. When the content has to be duplicated, it is recommended to avoid executing the expression again, and just access the cached result.

```
<?php

$current  = $array[$i];
$next     = $array[$i + 1];
$nextnext = $array[$i + 1]; // OOps, nextnext is wrong.

// Initialization
$previous = foo($array[1]); // previous is initialized with the first value on purpose
$next     = foo($array[1]); // the second call to foo() with the same arguments␣
↪should be avoided
// the above can be rewritten as :
$next     = $previous; // save the processing.

for($i = 1; $i < 200; ++$i) {
    $next = doSomething();
}
?>
```

Short name	Structures/IdenticalConsecutive
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Instant (5 mins)

9.249 Identical On Both Sides

Operands should be different when comparing or making a logical combination. Of course, the value each operand holds may be identical. When the same operand appears on both sides of the expression, the result is know before execution.

```
<?php

// Trying to confirm consistency
if ($login == $login) {
    doSomething();
}

// Works with every operators
if ($object->login( ) !== $object->login()) {
    doSomething();
}

if ($sum >= $sum) {
    doSomething();
}

//
```

(continues on next page)

(continued from previous page)

```
if ($mask && $mask) {
    doSomething();
}

if ($mask || $mask) {
    doSomething();
}

?>
```

9.249.1 Suggestions

- Remove one of the alternative, and remove the logical link
- Modify one of the alternative, and make it different from the other

Short name	Structures/IdenticalOnBothSides
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*phpMyAdmin*, *HuMo-Gen*

9.250 If With Same Conditions

Successive If / then structures that have the same condition may be either merged or have one of the condition changed.

```
<?php

if ($a == 1) {
    doSomething();
}

if ($a == 1) {
    doSomethingElse();
}

// May be replaced by
if ($a == 1) {
    doSomething();
    doSomethingElse();
}

?>
```

Note that if the values used in the condition have been modified in the first if/then structure, the two distinct conditions may be needed.

```
<?php

// May not be merged
if ($a == 1) {
    // Check that this is really the situation
```

(continues on next page)

(continued from previous page)

```
    $a = checkSomething();
}

if ($a == 1) {
    doSomethingElse();
}

?>
```

9.250.1 Suggestions

- Merge the two conditions so the condition is used once.
- Change one of the condition, so they are different
- Make it obvious that the first condition is a try, preparing the normal conditions.

Short name	Structures/IfWithSameConditions
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*phpMyAdmin*, *Phpdocumentor*

9.251 Iffectations

Affectations that appears in a condition.

Iffectations are a way to do both a test and an affectations. They may also be typos, such as if ($x = 3) { ... }, leading to a constant condition.

```
<?php

// an iffectation : assignation in a If condition
if($connexion = mysql_connect($host, $user, $pass)) {
    $res = mysql_query($connexion, $query);
}

// Iffectation may happen in while too.
while($row = mysql_fetch($res)) {
    $store[] = $row;
}

?>
```

9.251.1 Suggestions

- Move the assignation inside the loop, and make an existence test in the condition.
- Move the assignation before the if/then, make an existence test in the condition.

Short name	Structures/Iffectation
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

9.252 Illegal Name For Method

PHP has reserved usage of methods starting with __ for magic methods. It is recommended to avoid using this prefix, to prevent confusions.

```
<?php

class foo{
    // Constructor
    function __construct() {}

    // Constructor's typo
    function __constructor() {}

    // Illegal function name, even as private
    private function __bar() {}
}

?>
```

See also Magic Methods.

9.252.1 Suggestions

- Avoid method names starting with a double underscore : __
- Use method visibilities to ensure that methods are only available to the current class or its children

Short name	Classes/WrongName
Rulesets	*Analyze*
Severity	Major
Time To Fix	Slow (1 hour)
Examples	*PrestaShop*, *Magento*

9.253 Implement Is For Interface

With class heritage, implements should be used for interfaces, and extends with classes.

PHP defers the implements check until execution : the code in example does lint, but won,t run.

```
<?php

class x {
    function foo() {}
```

(continues on next page)

(continued from previous page)

```
}

interface y {
    function foo();
}

// Use implements with an interface
class z implements y {}

// Implements is for an interface, not a class
class z implements x {}

?>
```

9.253.1 Suggestions

- Create an interface from the class, and use it with the implements keyword

Short name	Classes/ImplementIsForInterface
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.254 Implemented Methods Are Public

Class methods that are defined in an interface must be public. They cannot be either private, nor protected.

This error is not reported by lint, but is reported at execution time.

```
<?php

interface i {
    function foo();
}

class X {
    // This method is defined in the interface : it must be public
    protected function foo() {}

    // other methods may be private
    private function bar() {}
}

?>
```

See also Interfaces and Interfaces - the next level of abstraction.

9.254.1 Suggestions

- Make the implemented method public

Short name	Classes/ImplementedMethodsArePublic
Rulesets	*Analyze*
Severity	Major
Time To Fix	Instant (5 mins)

9.255 Implied If

It is confusing to emulate if/then with boolean operators.

It is possible to emulate a if/then structure by using the operators 'and' and 'or'. Since optimizations will be applied to them : when the left operand of 'and' is false, the right one is not executed, as its result is useless; when the left operand of 'or' is true, the right one is not executed, as its result is useless;

However, such structures are confusing. It is easy to misread them as conditions, and ignore an important logic step.

```
<?php

// Either connect, or die
mysql_connect('localhost', $user, $pass) or die();

// Defines a constant if not found.
defined('SOME_CONSTANT') and define('SOME_CONSTANT', 1);

// Defines a default value if provided is empty-ish
// Warning : this is
$user = $_GET['user'] || 'anonymous';

?>
```

It is recommended to use a real 'if then' structures, to make the condition readable.

Short name	Structures/ImpliedIf
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)
ClearPHP	no-implied-if

9.256 Implode One Arg

implode() may be called with one arg. It is recommended to avoid it.

Using two arguments makes it less surprising to new comers, and consistent with explode() syntax.

```
<?php

$array = range('a', 'c');

// empty string is the glue
print implode('', $array);

// only the array : PHP uses the empty string as glue.
// Avoid this
```

(continues on next page)

(continued from previous page)

```
print implode($array);

?>
```

See also implode.

9.256.1 Suggestions

- Add an empty string as first argument

Short name	Php/ImplodeOneArg
Rulesets	*Suggestions*, *php-cs-fixable*
Severity	Minor
Time To Fix	Quick (30 mins)

9.257 Implode() Arguments Order

implode() accepted two signatures, but is only recommending one. Both types orders of string then array, and array then string have been possible until PHP 7.4.

In PHP 7.4, the order array then string is deprecated, and emits a warning. It will be removed in PHP 8.0.

```
<?php

$glue = ',';
$pieces = range(0, 4);

// documented argument order
$s = implode($glue, $pieces);

// Pre 7.4 argument order
$s = implode($pieces, $glue);

// both produces 0,1,2,3,4

?>
```

See also implode().

9.257.1 Suggestions

- Always use the array as the second argument

Short name	Structures/ImplodeArgsOrder
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.258 Inclusion Wrong Case

Inclusion should follow exactly the case of included files and path. This prevents the infamous case-sensitive filesystem bug, where files are correctly included in a case-insensitive system, and failed to be when moved to production.

```
<?php

// There must exist a path called path/to and a file library.php with this case
include path/to/library.php;

// Error on the case, while the file does exist
include path/to/LIBRARY.php;

// Error on the case, on the PATH
include path/TO/library.php;

?>
```

See also include_once, about case sensitivity and inclusions.

9.258.1 Suggestions

- Make the inclusion string identical to the file name.
- Change the name of the file to reflect the actual inclusion. This is the best way when a naming convention has been set up for the project, and the file doesn't adhere to it. Remember to change all other inclusion.

Short name	Files/InclusionWrongCase
Rulesets	*Analyze*
Severity	Major
Time To Fix	Slow (1 hour)

9.259 Incompatible Signature Methods

Methods should have the same signature when being overwritten.

The same signatures means the children class must have : + the same name + the same visibility or less restrictive + the same typehint or removed + the same default value or removed + a reference like its parent

This problem emits a fatal error, for abstract methods, or a warning error, for normal methods. Yet, it is difficult to lint, because classes are often stored in different files. As such, PHP do lint each file independently, as unknown parent classes are not checked if not present. Yet, when executing the code, PHP lint the actual code and may encounter a fatal error.

```
<?php

class a {
    public function foo($a = 1) {}
}

class ab extends a {
    // foo is overloaded and now includes a default value for $a
    public function foo($a) {}
```

(continues on next page)

(continued from previous page)

```
}

?>
```

See also Object Inheritance.

9.259.1 Suggestions

- Make signatures compatible again

Short name	Classes/IncompatibleSignature
Rulesets	*Analyze*, *LintButWontExec*
Php Version	7.4-
Severity	Critical
Time To Fix	Quick (30 mins)
Examples	*SuiteCrm*

9.260 Incompatible Signature Methods With Covariance

Methods should have the compatible signature when being overwritten.

The same signatures means the children class must have : + the same name + the same visibility or less restrictive + the same contravariant typehint or removed + the same covariant return typehint or removed + the same default value or removed + a reference like its parent

This problem emits a fatal error, for abstract methods, or a warning error, for normal methods. Yet, it is difficult to lint, because classes are often stored in different files. As such, PHP do lint each file independently, as unknown parent classes are not checked if not present. Yet, when executing the code, PHP lint the actual code and may encounter a fatal error.

```
<?php

class a {
    public function foo($a = 1) {}
}

class ab extends a {
    // foo is overloaded and now includes a default value for $a
    public function foo($a) {}
}

?>
```

See also Object Inheritance, PHP RFC: Covariant Returns and Contravariant Parameters and *Incompatible Signature Methods*.

9.260.1 Suggestions

- Make signatures compatible again

Short name	Classes/IncompatibleSignature74
Rulesets	*Analyze*
Php Version	7.4+
Severity	Critical
Time To Fix	Quick (30 mins)
Examples	*SuiteCrm*

9.261 Incompilable Files

Files that cannot be compiled, and, as such, be run by PHP. Scripts are linted against various versions of PHP.

This is usually undesirable, as all code must compile before being executed. It may be that such files are not compilable because they are not yet ready for an upcoming PHP version.

```
<?php

// Can't compile this : Print only accepts one argument
print $a, $b, $c;

?>
```

Code that is not compilable with older PHP versions means that the code is breaking backward compatibility : good or bad is project decision.

When the code is used as a template for PHP code generation, for example at installation time, it is recommended to use a distinct file extension, so as to distinguish them from actual PHP code.

9.261.1 Suggestions

- If this file is a template for PHP code, change the extension to something else than .php
- Fix the syntax so it works with various versions of PHP

Short name	Php/Incompilable
Rulesets	*Analyze*
Severity	Critical
Time To Fix	Slow (1 hour)
ClearPHP	no-incompilable
Examples	*xataface*

9.262 Inconsistent Elseif

Chaining if/elseif requires a consistent string of conditions. The conditions are executed one after the other, and the conditions shouldn't overlap.

This analysis reports chains of elseif that don't share a common variable (or array, or property, etc..). As such, testing different conditions are consistent.

```php
<?php

// $a is always common, so situations are mutually exclusive
if ($a === 1) {
    doSomething();
} else if ($a > 1) {
    doSomethingElse();
} else {
    doSomethingDefault();
}

// $a is always common, so situations are mutually exclusive
// although, it may be worth checking the consistency here
if ($a->b === 1) {
    doSomething();
} else if ($a->c > 1) {
    doSomethingElse();
} else {
    doSomethingDefault();
}

// if $a === 1, then $c doesn't matter?
// This happens, but then logic doesn't appear in the code.
if ($a === 1) {
    doSomething();
} else if ($c > 1) {
    doSomethingElse();
} else {
    doSomethingDefault();
}

?>
```

Short name	Structures/InconsistentElseif
Rulesets	*Analyze*
Severity	Major
Time To Fix	Slow (1 hour)

9.263 Indices Are Int Or String

Indices in an array notation such as `$array['indice']` may only be integers or string.

Boolean, Null or float will be converted to their integer or string equivalent.

```php
<?php
    $a = [true => 1,
          1.0  => 2,
          1.2  => 3,
          1    => 4,
          '1'  => 5,
          0.8  => 6,
          0x1  => 7,
          01   => 8,
```

(continues on next page)

(continued from previous page)

```
          null  => 1,
          ''    => 2,

          false => 1,
          0     => 2,

          '0.8' => 3,
          '01'  => 4,
          '2a'  => 5
          ];

    print_r($a);

/*
The above displays
Array
(
    [1] => 8
    [0] => 2
    [] => 2
    [0.8] => 3
    [01] => 4
    [2a] => 5
)
*/
?>
```

Decimal numbers are rounded to the closest integer; Null is transtyped to ‘’ (empty string); true is 1 and false is 0; Integers in strings are transtyped, while partial numbers or decimals are not analyzed in strings.

As a general rule of thumb, only use integers or strings that don’t look like integers.

This analyzer may find constant definitions, when available.

Note also that PHP detects integer inside strings, and silently turn them into integers. Partial and octal numbers are not transformed.

```
<?php
    $a = [1      => 1,
          '2'    => 2,
          '011'  => 9, // octal number
          '11d'  => 11, // partial number
          ];

    var_dump($a);

/*
The above displays
array(4) {
  [1]=>
  int(1)
  [2]=>
  int(2)
  [011]=>
  int(9)
  [11d]=>
  int(11)
```

(continues on next page)

(continued from previous page)

```
}*/
?>
```

See also Arrays syntax.

9.263.1 Suggestions

- Do not use any type but string or integer
- Force typecast the keys when building an array

Short name	Structures/IndicesAreIntOrString
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Zencart*, *Mautic*

9.264 Indirect Injection

Look for injections through indirect usage for GPRC values ($_GET, $_POST, $_REQUEST, $_COOKIE).

```
<?php

$a = $_GET['a'];
echo $a;

function foo($b) {
    echo $b;
}
foo($_POST['c']);

?>
```

9.264.1 Suggestions

- Always validate incoming values before using them.

Short name	Security/IndirectInjection
Rulesets	*Security*
Severity	Critical
Time To Fix	Slow (1 hour)

9.265 Infinite Recursion

A method is calling itself, with unchanged arguments. This will probably repeat indefinitely.

This applies to recursive functions without any condition. This also applies to function which inject the incoming arguments, without modifications.

```
<?php

function foo($a, $b) {
    if ($a > 10) {
        return;
    }
    foo($a, $b);
}

function foo2($a, $b) {
    ++$a;   // $a is modified
    if ($a > 10) {
        return;
    }
    foo2($a, $b);
}

?>
```

9.265.1 Suggestions

- Modify arguments before injecting them again in the same method
- Use different values when calling the same method

Short name	Structures/InfiniteRecursion
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)

9.266 Instantiating Abstract Class

PHP cannot instantiate an abstract class.

The classes are actually abstract classes, and should be derived into a concrete class to be instantiated.

```
<?php

abstract class Foo {
    protected $a;
}

class Bar extends Foo {
    protected $b;
}

// instantiating a concrete class.
new Bar();

// instantiating an abstract class.
// In real life, this is not possible also because the definition and the␣
↪instantiation are in the same file
new Foo();
```

(continues on next page)

(continued from previous page)

```
?>
```

See also Class Abstraction.

Short name	Classes/InstantiatingAbstractClass
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)

9.267 Insufficient Property Typehint

The typehint used for a class property doesn't cover all it usage.

The typehint is insufficient when a undefined method is called, or if members are access while the typehint is an interface.

```
<?php

class A {
    function a1() {}
}

// PHP 7.4 and more recent
class B {
    private A $a = null;

    function b2() {
        // this method is available in A
        $this->a->a1();
        // this method is NOT available in A
        $this->a->a2();
    }
}

// Supported by all PHP versions
class C {
    private $a = null;

    function __construct(A $a) {
        $this->a = $a;
    }

    function b2() {
        // this method is available in A
        $this->a->a1();
        // this method is NOT available in A
        $this->a->a2();
    }
}

?>
```

This analysis relies on typehinted properties, as introduced in PHP 7.4. It also relies on typehinted assignations at construct time : the typehint of the assigned argument will be used as the property typehint. Getters and setters are not considered here.

9.267.1 Suggestions

- Change the typehint to match the actual usage of the object in the class.

Short name	Classes/InsufficientPropertyTypehint
Rulesets	*ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.268 Insufficient Typehint

An argument is typehinted, but it actually calls methods that are not listed in the interface.

Classes may be implementing more methods than the one that are listed in the interface they also implements. This means that filtering objects with a typehint, but calling other methods will be solved at execution time : if the method is available, it will be used; if it is not, a fatal error is reported.

```php
<?php

class x implements i {
    function methodI() {}
    function notInI() {}
}

interface i {
    function methodI();
}

function foo(i $x) {
    $x->methodI(); // this call is valid
    $x->notInI();  // this call is not garanteed
}
?>
```

Inspired by discussion with Brandon Savage.

9.268.1 Suggestions

- Extend the interface with the missing called methods
- Change the body of the function to use only the methods that are available in the interface
- Change the used objects so they don't depend on extra methods

Short name	Functions/InsufficientTypehint
Rulesets	*Analyze*, *Typechecks*
Severity	Major
Time To Fix	Quick (30 mins)

9.269 Integer As Property

It is backward incompatible to use integers are property names. This feature was introduced in PHP 7.2.

If the code must be compatible with previous versions, avoid casting arrays to object.

```
<?php

// array to object
$arr = [0 => 1];
$obj = (object) $arr;
var_dump(
    $obj,
    $obj->{'0'}, // PHP 7.2+ accessible
    $obj->{0} // PHP 7.2+ accessible

    $obj->{'b'}, // always been accessible
);
?>
```

See also PHP RFC: Convert numeric keys in object/array casts.

Short name	Classes/IntegerAsProperty
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.2 and more recent
Severity	Major
Time To Fix	Slow (1 hour)

9.270 Integer Conversion

Comparing incoming variables to integer may lead to injection.

When comparing a variable to an integer, PHP applies type juggling, and transform the variable in an integer too. When the value converts smoothly to an integer, this means the validation may pass and yet, the value may carry an injection.

```
<?php

// This is safer
if ($_GET['x'] === 2) {
    echo $_GET['x'];
}

// Using (int) for validation and display
if ((int) $_GET['x'] === 2) {
    echo (int) $_GET['x'];
}

// This is an injection
if ($_GET['x'] == 2) {
```

(continues on next page)

(continued from previous page)

```
    echo $_GET['x'];
}

// This is unsafe, as $_GET['x']  is tester as an integer, but echo'ed raw
if ((int) $_GET['x'] === 2) {
    echo $_GET['x'];
}

?>
```

This analysis spots situations where an incoming value is compared to an integer. The usage of the validated value is not considered.

See also Type Juggling Authentication Bypass Vulnerability in CMS Made Simple, PHP STRING COMPARISON VULNERABILITIES and PHP Magic Tricks: Type Juggling.

9.270.1 Suggestions

-

Short name	Security/IntegerConversion
Rulesets	*Security*
Severity	Major
Time To Fix	Quick (30 mins)

9.271 Interfaces Don't Ensure Properties

When using an interface as a typehint, properties are not enforced, nor available.

An interface is a template for a class, which specify the minimum amount of methods and constants. Properties are never defined in an interface, and should not be relied upon.

```
<?php

interface i {
    function m () ;
}

class x implements i {
    public $p = 1;

    function m() {
        return $this->p;
    }
}

function foo(i $i, x $x) {
    // this is invalid, as $p is not defined in i, so it may be not available
    echo $i->p;

    // this is valid, as $p is defined in $x
    echo $x->p;
```

(continues on next page)

(continued from previous page)

```
}

?>
```

9.271.1 Suggestions

- Use classes for typehint when properties are accessed
- Only use methods and constants which are available in the interface

Short name	Interfaces/NoGaranteeForPropertyConstant
Rulesets	*Analyze*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.272 Interfaces Is Not Implemented

Classes that implements interfaces, must implements each of the interface's methods.

```
<?php

class x implements i {
    // This method implements the foo method from the i interface
    function foo() {}

    // The method bar is missing, yet is requested by interface i
    function foo() {}
}

interface i {
    function foo();
    function bar();
}

?>
```

This problem tends to occur in code that splits interfaces and classes by file. This means that PHP's linting will skip the definitions and not find the problem. At execution time, the definitions will be checked, and a Fatal error will occur.

This situation usually detects code that was forgotten during a refactorisation of the interface or the class and its sibblings.

See also Interfaces.

9.272.1 Suggestions

- Implements all the methods from the interfaces
- Remove the class
- Make the class abstract

- Make the missing methods abstract

Short name	Interfaces/IsNotImplemented
Rulesets	*Analyze*, *ClassReview*, *LintButWontExec*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.273 Interpolation

The following strings contain variables that are will be replaced. However, the following characters are ambiguous, and may lead to confusion.

```
<?php

class b {
    public $b = 'c';
    function __toString() { return __CLASS__; }
}
$x = array(1 => new B());

// -> after the $x[1] looks like a 2nd dereferencing, but it is not.
print $x[1]->b;
// displays : b->b

print {$x[1]->b};
// displays : c

?>
```

It is advised to add curly brackets around those structures to make them non-ambiguous.

See also Double quoted.

Short name	Type/StringInterpolation
Rulesets	*Coding Conventions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.274 Invalid Constant Name

There is a naming convention for PHP constants names.

According to PHP's manual, constant names, ' A valid constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.'.

Constant, must follow this regex : `/[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*/`.

In particular when defined using define() function, no error is produced. When using `const`, on the other hand, the

```
<?php

define('+3', 1); // wrong constant!

```

(continues on next page)

(continued from previous page)

```
echo constant('+3'); // invalid constant access

?>
```

See also Constants.

9.274.1 Suggestions

- Change constant name

Short name	Constants/InvalidName
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*OpenEMR*

9.275 Invalid Octal In String

Any octal sequence inside a string can’t be go 7. Those will be a fatal error at parsing time.

The check is applied to the string, starting with PHP 7.1. In PHP 7.0 and older, those sequences were silently adapted (modulo/% 0).

```
<?php

// A valid octal in a PHP string
echo 0; // @

// Emit a warning in PHP 7.1
//Octal escape sequence overflow 0 is greater than 7
echo 0; // @

// Silent conversion
echo 8; // 8

?>
```

See also Integers.

9.275.1 Suggestions

- Use a double slash to avoid the sequence to be an octal sequence
- Use a function call, such as decoct() to convert larger number to octal notation

Short name	Type/OctalInString
Rulesets	*CompatibilityPHP71*
Php Version	With PHP 7.1 and older
Severity	Major
Time To Fix	Quick (30 mins)

9.276 Invalid Pack Format

Some characters are invalid in a pack() format string.

pack() and unpack() accept the following format specifiers : `aAhHcCsSnviIlLNVqQJPfgGdeExXZ`.

unpack() also accepts a name after the format specifier and an optional quantifier.

All other situations is not a valid, and produces a warning : `pack(): Type t: unknown format code`

```
<?php
    $binarydata = pack(nvc*, 0x1234, 0x5678, 65, 66);

    // the first unsigned short is stored as 'first'. The next matches are names with␣
↪numbers.
    $res = unpack('nfirst/vc*', $binarydata);
?>
```

Check pack() documentation for format specifiers that were introduced in various PHP version, namely 7.0, 7.1 and 7.2.

See also pack and unpack.

9.276.1 Suggestions

- Fix the packing format with correct values

Short name	Structures/InvalidPackFormat
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)

9.277 Invalid Regex

The PCRE regex doesn't compile. It isn't a valid regex.

Several reasons may lead to this situation : syntax error, Unknown modifier, missing parenthesis or reference.

```
<?php

// valid regex
preg_match('/[abc]/', $string);

// invalid regex (missing terminating ] for character class
preg_match('/[abc/', $string);

?>
```

Regex are check with the Exakat version of PHP.

Dynamic regex are only checked for simple values. Dynamic values may eventually generate a compilation error.

9.277.1 Suggestions

- Fix the regex before running it

Short name	Structures/InvalidRegex
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*SugarCrm*

9.278 Is Actually Zero

This addition actually may be simplified because one term is actually negated by another.

This kind of error happens when the expression is very large : the more terms are included, the more chances are that some auto-annihilation happens.

This error may also be a simple typo : for example, calculating the difference between two consecutive terms.

```
<?php

// This is quite obvious
$a = 2 - 2;

// This is obvious too. This may be a typo-ed difference between two consecutive␣
↪terms.
// Could have been $c = $fx[3][4] - $fx[3][3] or $c = $fx[3][5] - $fx[3][4];
$c = $fx[3][4] - $fx[3][4];

// This is less obvious
$a = $b[3] - $c + $d->foo(1,2,3) + $c + $b[3];

?>
```

9.278.1 Suggestions

- Clean the code and remove the null sum
- Fix one of the variable : this expression needs another variable here
- When adding differences, calculate the difference in a temporary variable first.

Short name	Structures/IsZero
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Dolibarr*, *SuiteCrm*

9.279 Is_A() With String

When using is_a() with a string as first argument, the third argument is compulsory.

```
<?php

// is_a() works with string as first argument, when the third argument is 'true'
if (is_s('A', 'B', true)) {}

// is_a() works with object as first argument
if (is_s(new A, 'A')) {}
?>
```

See also is_a().

9.279.1 Suggestions

- Add the third argument, and set it to true
- Use an object as a first argument

Short name	Php/IsAWithString
Rulesets	*Analyze*, *Rector*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.280 Isset Multiple Arguments

isset() may be used with multiple arguments and acts as a AND.

```
<?php

// isset without and
if (isset($a, $b, $c)) {
    // doSomething()
}

// isset with and
if (isset($a) && isset($b) && isset($c)) {
    // doSomething()
}

?>
```

See also Isset <http://www.php.net/‘isset>‘_.

9.280.1 Suggestions

- Merge all isset() calls into one

Short name	Php/IssetMultipleArgs
Rulesets	*Suggestions*, *php-cs-fixable*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*ThinkPHP*, *LiveZilla*

9.281 Isset() On The Whole Array

Isset() works quietly on a whole array. There is no need to test all previous index before testing for the target index.

```
<?php

// Straight to the point
if (isset($a[1]['source'])) {
    // Do something with $a[1]['source']
}

// Doing too much work
if (isset($a) && isset($a[1]) && isset($a[1]['source'])) {
    // Do something with $a[1]['source']
}

?>
```

There is a gain in readability, by avoiding long and hard to read logical expression, and reducing them in one simple isset call.

There is a gain in performances by using one call to isset, instead of several, but it is a micro-optimization.

See also Isset <http://www.php.net/‘isset>‘_.

9.281.1 Suggestions

- Remove all unnecessary calls to isset()

Short name	Performances/IssetWholeArray
Rulesets	*Suggestions*, *Performances*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Tine20*, *ExpressionEngine*

9.282 Joining file()

Use file() to read lines separately.

Applying join(‘’,) or implode(‘’,) to the result of file() provides the same results than using file_get_contents(), but at a higher cost of memory and processing.

If the delimiter is not ‘’, then implode() and file() are a better solution than file_get_contents() and str_replace() or nl2br().

```
<?php

// memory intensive
$content = file_get_contents('path/to/file.txt');

// memory and CPU intensive
$content = join('', file('path/to/file.txt'));
```

(continues on next page)

(continued from previous page)

```
// Consider reading the data line by line and processing it along the way,
// to save memory
$fp = fopen('path/to/file.txt', 'r');
while($line = fget($fp)) {
    // process a line
}
fclose($fp);

?>
```

Always use file_get_contents() to get the content of a file as a string. Consider using readfile() to echo the content directly to the output.

See also file_get_contents and file.

9.282.1 Suggestions

- Use file_get_contents() instead of implode(file()) to read the whole file at once.
- Use readfile() to echo the content to stdout at once.
- Use fopen() to read the lines one by one, generator style.

Short name	Performances/JoinFile
Rulesets	*Performances*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*WordPress*, *SPIP*, *ExpressionEngine*, *PrestaShop*

9.283 Keep Files Access Restricted

Avoid using 0777 as file or directory mode. In particular, setting a file or a directory to 0777 (or universal read-write-execute) may lead to security vulnerabilities, as anything on the server may read, write and even execute

File mode may be changed using the chmod() function, or at directory creation, with mkdir().

```
<?php

file_put_contents($file, $content);

// this file is accessible to the current user, and to his group, for reading and␣
↪writing.
chmod($file, 0550);

// this file is accessible to everyone
chmod($file, 0777);

?>
```

By default, this analysis report universal access (0777). It is possible to make this analysis more restrictive, by providing more forbidden modes in the `filePrivileges` parameter. For example : `511,510,489`. Only use a decimal representation.

See also *Mkdir Default* and Least Privilege Violation.

9.283.1 Suggestions

- Set the file mode to a level of restriction as low as possible.

Name	Default	Type	Description
filePrivileges	0777	string	List of forbidden file modes (comma separated).

Short name	Security/KeepFilesRestricted
Rulesets	*Security*
Severity	Minor
Time To Fix	Quick (30 mins)

9.284 Large Try Block

Try block should enclosing only the expression that may emit an exception.

When writing large blocks of code in a try, it becomes difficult to understand where the expression is coming from. Large blocks may also lead to catch multiples exceptions, with a long list of catch clause.

In particular, the catch clause will resume the execution without knowing where the try was interrupted : there are no indication of achievement, even partial. In fact, catching an exception signals a very dirty situation.

```php
<?php

// try is one expression only
try {
    $database->query($query);
} catch (DatabaseException $e) {
    // process exception
}

// Too many expressions around the one that may actually emit the exception
try {
    $SQL = build_query($arguments);
    $database = new Database($dsn);
    $database->setOption($options);
    $statement = $database->prepareQuery($SQL);
    $result = $statement->query($query);
} catch (DatabaseException $e) {
    // process exception
}

?>
```

This analysis reports try blocks that are 5 lines or more. This threshold may be configured with the directive `tryBlockMaxSize`. Catch clause, and finally are not considered here.

9.284.1 Suggestions

- Reduce the amount of code in the block, by moving it before and after

Name	Default	Type	Description
tryBlockMaxSize	5	integer	Maximal number of expressions in the try block.

Short name	Exceptions/LargeTryBlock
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.285 List Short Syntax

Usage of short syntax version of list().

```
<?php

// PHP 7.1 short list syntax
// PHP 7.1 may also use key => value structures with list
[$a, $b, $c] = ['2', 3, '4'];

// PHP 7.0 list syntax
list($a, $b, $c) = ['2', 3, '4'];

?>
```

Short name	Php/ListShortSyntax
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.1 and more recent
Severity	Major
Time To Fix	Quick (30 mins)

9.286 List With Appends

List() behavior has changed in PHP 7.0 and it has impact on the indexing when list is used with the [] operator.

```
<?php

$x = array();
list($x[], $x[], $x[]) = [1, 2, 3];

print_r($x);

?>
```

In PHP 7.0, results are ::

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
)
```

In PHP 5.6, results are ::

```
Array
(
    [0] => 3
    [1] => 2
    [2] => 1
)
```

9.286.1 Suggestions

- Refactor code to avoid using append in a list() call

Short name	Php/ListWithAppends
Rulesets	*CompatibilityPHP70*
Severity	Minor
Time To Fix	Slow (1 hour)

9.287 List With Keys

Setting keys when using list() is a PHP 7.1 feature.

```
<?php

// PHP 7.1 and later only
list('a' => $a, 'b' => $b) = ['b' => 1, 'c' => 2, 'a' => 3];

?>
```

Short name	Php/ListWithKeys
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.1 and more recent
Severity	Major
Time To Fix	Quick (30 mins)

9.288 List With Reference

Support for references in list calls is not backward compatible with older versions of PHP. The support was introduced in PHP 7.3.

```
<?php

$array = [1,2,3];

[$c, &$d, $e] = $a;
```

(continues on next page)

(continued from previous page)

```
$d++;
$c++;
print_r($array);
/*
displays
Array
(
    [0] => 1  // Not a reference to $c, unchanged
    [1] => 3  // Reference from $d
    [2] => 3
)
*/
?>
```

See also list() Reference Assignment.

9.288.1 Suggestions

- Avoid using references in list for backward compatibility

Short name	Php/ListWithReference
Rule-sets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.3 and more recent
Severity	Major
Time To Fix	Slow (1 hour)

9.289 Locally Unused Property

Those properties are defined in a class, and this class doesn't have any method that makes use of them.

While this is syntactically correct, it is unusual that defined resources are used in a child class. It may be worth moving the definition to another class, or to move accessing methods to the class.

```
<?php

class foo {
    public $unused, $used;// property $unused is never used in this class

    function bar() {
        $this->used++; // property $used is used in this method
    }
}

class foofoo extends foo {
    function bar() {
        $this->unused++; // property $unused is used in this method, but defined in
↪the parent class
```

(continues on next page)

(continued from previous page)

```
    }
}

?>
```

9.289.1 Suggestions

- Move the property definition to the child classes
- Move some of the child method, using the property, to the parent class

Short name	Classes/LocallyUnusedProperty
Rulesets	*Dead code*
Severity	Minor
Time To Fix	Slow (1 hour)

9.290 Logical Mistakes

Avoid logical mistakes within long expressions.

Sometimes, the logic is not what it seems. It is important to check the actual impact of every part of the logical expression. Do not hesitate to make a table with all possible cases. If those cases are too numerous, it may be time to rethink the whole expression.

```
<?php

// Always true
if ($a != 1 || $a != 2) { }

// $a == 1 is useless
if ($a == 1 || $a != 2) {}

// Always false
if ($a == 1 && $a == 2) {}

// $a != 2 is useless
if ($a == 1 && $a != 2) {}

?>
```

Based on article from `Andrey Karpov` Logical Expressions in C/C++. Mistakes Made by Professionals

9.290.1 Suggestions

- Change the expressions for them to have a real meaning

Short name	Structures/LogicalMistakes
Rulesets	*Analyze*, *CI-checks*
Severity	Critical
Time To Fix	Quick (30 mins)
Examples	*Dolibarr*, *Cleverstyle*

9.291 Logical Operators Favorite

PHP has two sets of logical operators : letters (and, or, xor) and chars (&&, ||, ^).

The analyzed code has less than 10% of one of the two sets : for consistency reasons, it is recommended to make them all the same.

Warning : the two sets of operators have different precedence levels. Using and or && is not exactly the same, especially and not only, when assigning the results to a variable.

```
<?php

$a1 = $b and $c;
$a1 = $b and $c;
$a1 = $b and $c;
$a1 = $b or $c;
$a1 = $b OR $c;
$a1 = $b and $c;
$a1 = $b and $c;
$a1 = $b and $c;
$a1 = $b or $c;
$a1 = $b OR $c;
$a1 = $b ^ $c;

?>
```

Using and or && are also the target of other analysis.

See also Logical Operators and Operators Precedence.

9.291.1 Suggestions

- Pick a favorite, and enforce it

Short name	Php/LetterCharsLogicalFavorite
Rulesets	*Top10*

9.292 Logical Should Use Symbolic Operators

Logical operators come in two flavors : and / &&, || / or, ^ / xor. However, they are not exchangeable, as && and and have different precedence.

```
<?php

// Avoid lettered operator, as they have lower priority than expected
$a = $b and $c;
// $a === 3 because equivalent to ($a = $b) and $c;

// safe way to write the above :
$a = ($b and $c);

$a = $b && $c;
// $a === 1
```

(continues on next page)

(continued from previous page)

```

?>
```

It is recommended to use the symbol operators, rather than the letter ones.

See also Logical Operators.

9.292.1 Suggestions

- Change the letter operators to the symbol one : and => &&, or => ||, xor => ^. Review the new expressions as processing order may have changed.
- Add parenthesis to make sure that the order is the expected one

Short name	Php/LogicalInLetters
Rulesets	*Analyze*, *Suggestions*, *Top10*, *php-cs-fixable*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
ClearPHP	no-letter-logical
Examples	*Cleverstyle*, *OpenConf*

9.293 Logical To in_array

Multiples exclusive comparisons may be replaced by in_array().

in_array() makes the alternatives more readable, especially when the number of alternatives is large. In fact, the list of alternative may even be set in a variable, and centralized for easier management.

Even two 'or' comparisons are slower than using a in_array() call. More calls are even slower than just two. This is a micro-optimisation : speed gain is low, and marginal. Code centralisation is a more significant advantage.

```
<?php

// Set the list of alternative in a variable, property or constant.
$valid_values = array(1, 2, 3, 4);
if (in_array($a, $valid_values) ) {
    // doSomething()
}

if ($a == 1 || $a == 2 || $a == 3 || $a == 4) {
    // doSomething()
}

// in_array also works with strict comparisons
if (in_array($a, $valid_values, true) ) {
    // doSomething()
}

if ($a === 1 || $a === 2 || $a === 3 || $a === 4) {
    // doSomething()
}

?>
```

See also in_array().

9.293.1 Suggestions

- Replace the list of comparisons with a in_array() call on an array filled with the various values
- Replace the list of comparisons with a isset() call on a hash whose keys are the various values

Short name	Performances/LogicalToInArray
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high
Examples	*Zencart*

9.294 Lone Blocks

Any grouped code without a commanding structure is useless.

Blocks are compulsory when defining a structure, such as a class or a function. They are most often used with flow control instructions, like if then or switch.

Blocks are also valid syntax that group several instructions together, though they have no effect at all, except confuse the reader. Most often, it is a ruin from a previous flow control instruction, whose condition was removed or commented. They should be removed.

```php
<?php

    // Lone block
    //foreach($a as $b)
    {
        $b++;
    }
?>
```

9.294.1 Suggestions

- Remove the useless curly brackets

Short name	Structures/LoneBlock
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*ThinkPHP*, *Tine20*

9.295 Long Arguments

Long arguments should be put in variable, to preserve readability.

When literal arguments are too long, they break the hosting structure by moving the next argument too far on the right. Whenever possible, long arguments should be set in a local variable to keep the readability.

```
<?php

// Now the call to foo() is easier to read.
$reallyBigNumber = <<<BIGNUMBER
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
BIGNUMBER
foo($reallyBigNumber, 2, '12345678901234567890123456789012345678901234567890');

// where are the next arguments ?
foo(
↪'12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678
↪', 2, '12345678901234567890123456789012345678901234567890');

// This is still difficult to read
foo(<<<BIGNUMBER
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
BIGNUMBER
, 2, '12345678901234567890123456789012345678901234567890');

?>
```

Literal strings and heredoc strings, including variables, that are over 50 chars longs are reported here.

9.295.1 Suggestions

- Put the long arguments in a separate variable, and use the variable in the second expression, reducing its total length

Name	Default	Type	Description
codeTooLong	100	integer	Minimum size of a functioncall or a methodcall to be considered too long.

Short name	Structures/LongArguments
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Cleverstyle*, *Contao*

9.296 Lost References

Either avoid references, or propagate them correctly.

When assigning a referenced variable with another reference, the initial reference is lost, while the intend was to transfer the content.

```
<?php

function foo(&$lostReference, &$keptReference)
{
    $c = 'c';
```

(continues on next page)

(continued from previous page)

```
    // $lostReference was a reference, but now, it is another
    $lostReference =& $c;
    // $keptReference was a reference : now it contains the actual value
    $keptReference = $c;
}

$bar = 'bar';
$bar2 = 'bar';
foo($bar, $bar2);

//displays bar c, instead of bar bar
print $bar. ' '.$bar2;

?>
```

Do not reassign a reference with another reference. Assign new content to the reference to change its value.

9.296.1 Suggestions

- Always assign new value to an referenced argument, and don't reassign a new reference

Short name	Variables/LostReferences
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*WordPress*

9.297 Magic Visibility

The class magic methods must have public visibility and cannot be static.

```
<?php

class foo{
    // magic method must bt public and non-static
    public static function __clone($name) {     }

    // magic method can't be private
    private function __get($name) {     }

    // magic method can't be protected
    private function __set($name, $value) {     }

    // magic method can't be static
    public static function __isset($name) {     }
}

?>
```

See also Magic methods.

Short name	Classes/toStringPss
Rulesets	*CompatibilityPHP70*
Php Version	With PHP 5.4 and older
Severity	Major
Time To Fix	Quick (30 mins)

9.298 Make Global A Property

Calling global (or $GLOBALS) in methods is slower and less testable than setting the global to a property, and using this property.

Using properties is slightly faster than calling global or $GLOBALS, though the gain is not important.

Setting the property in the constructor (or in a factory), makes the class easier to test, as there is now a single point of configuration.

```php
<?php

// Wrong way
class fooBad {
    function x() {
        global $a;
        $a->do();
        // Or $GLOBALS['a']->do();
    }
}

class fooGood {
    private $bar = null;

    function __construct() {
        global $bar;
        $this->bar = $bar;
        // Even better, do this via arguments
    }

    function x() {
        $this->a->do();
    }
}

?>
```

9.298.1 Suggestions

- Avoid using global variables, and use properties instead
- Remove the usage of these global variables

Short name	Classes/MakeGlobalAProperty
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)

9.299 Make Magic Concrete

Speed up execution by replacing magic calls by concrete properties.

Magic properties are managed dynamically, with __get``and ``__set. They replace property access by a methodcall, and they are much slower than the first.

When a property name is getting used more often, it is worth creating a concrete property, and skip the method call. The threshold for 'magicMemberUsage' is 1, by default.

```
<?php

class x {
    private $values = array('a' => 1,
                            'b' => 2);

    function __get($name) {
        return $this->values[$name] ?? '';
    }
}

$x = new x();
// Access to 'a' is repeated in the code, at least 'magicMemberUsage' time (cf␣
↪configuration below)
echo $x->a;

?>
```

See also *Memoize MagicCall*.

9.299.1 Suggestions

- Make frequently used properties concrete; keep the highly dynamic as magic

Name	Default	Type	Description
magicMemberUsage	1	integer	Minimal number of magic member usage across the code, to trigger a concrete property.

Short name	Classes/MakeMagicConcrete
Rulesets	*Performances*
Severity	Minor
Time To Fix	Quick (30 mins)

9.300 Make One Call With Array

Avoid calling the same function several times by batching the calls with arrays.

Calling the same function to chain modifications tends to be slower than calling the same function with all the transformations at the same time. Some PHP functions accept scalars or arrays, and using the later is more efficient.

```
<?php

$string = 'abcdef';

//str_replace() accepts arrays as arguments
$string = str_replace( ['a', 'b', 'c'],
                       ['A', 'B', 'C'],
                       $string);

// Too many calls to str_replace
$string = str_replace( 'a', 'A', $string);
$string = str_replace( 'b', 'B', $string);
$string = str_replace( 'c', 'C', $string);

// Too many nested calls to str_replace
$string = str_replace( 'a', 'A', str_replace( 'b', 'B', str_replace( 'c', 'C',
↪$string)));

?>
```

Potential replacements :

Function	Replacement
str_replace() str_ireplace() substr_replace() preg_replace() preg_replace_callback()	str_replace() str_replace() substr_replace() preg_replace() preg_replace_callback_array()

```
<?php
$subject = 'Aaaaaa Bbb';


//preg_replace_callback_array() is better than multiple preg_replace_callback :
preg_replace_callback_array(
    [
        '~[a]+~i' => function ($match) {
            echo strlen($match[0]), ' matches for a found', PHP_EOL;
        },
        '~[b]+~i' => function ($match) {
            echo strlen($match[0]), ' matches for b found', PHP_EOL;
        }
    ],
    $subject
);

$result = preg_replace_callback('~[a]+~i', function ($match) {
            echo strlen($match[0]), ' matches for a found', PHP_EOL;
        }, $subject);

$result = preg_replace_callback('~[b]+~i', function ($match) {
            echo strlen($match[0]), ' matches for b found', PHP_EOL;
        }, $subject);

//str_replace() accepts arrays as arguments
$string = str_replace( ['a', 'b', 'c'],
                       ['A', 'B', 'C'],
                       $string);
```

(continues on next page)

(continued from previous page)

```
// Too many calls to str_replace
$string = str_replace( 'a', 'A');
$string = str_replace( 'b', 'B');
$string = str_replace( 'c', 'C');

?>
```

9.300.1 Suggestions

- use str_replace() with arrays as arguments.
- use preg_replace() with arrays as arguments.
- use preg_replace_callback() for merging multiple complex calls.

Short name	Performances/MakeOneCall
Rulesets	*Performances*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*HuMo-Gen*, *Edusoho*

9.301 Malformed Octal

Those numbers starts with a 0, so they are using the PHP octal convention. Therefore, one can't use 8 or 9 figures in those numbers, as they don't belong to the octal base. The resulting number will be truncated at the first erroneous figure. For example, 090 is actually 0, and 02689 is actually 22.

```
<?php

// A long way to write 0 in PHP 5
$a = 0890;

// A fatal error since PHP 7

?>
```

Also, note that very large octal, usually with more than 21 figures, will be turned into a real number and undergo a reduction in precision.

See also Integers.

Short name	Type/MalformedOctal
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and older
Severity	Minor
Time To Fix	Instant (5 mins)

9.302 Max Level Of Nesting

Avoid nesting structures too deep, as it hurts readability.

Nesting structures are : if/then, switch, for, foreach, while, do…while. Ternary operator, try/catch are not considered a nesting structures.

Closures, and more generally, functions definitions are counted separatedly.

This analysis checks for 4 levels of nesting, by default. This may be changed by configuration.

```php
<?php

// 5 levels of indentation
function foo() {
    if (1) {
        if (2) {
            if (3) {
                if (4) {
                    if (5) {
                        51;
                    } else {
                        5;
                    }
                } else {
                    4;
                }
            } else {
                3;
            }
        } else {
            2;
        }
    } else {
        1;
    }
}

// 2 levels of indentation
function foo() {
    if (1) {
        if (2) {
            // 3 levels of indentation
            return function () {
                if (3) {
                    if (4) {
                        if (5) {
                            51;
                        } else {
                            5;
                        }
                    } else {
                        4;
                    }
                } else {
                    3;
                }
            }
        } else {
            2;
        }
    } else {
        1;
```

(continues on next page)

(continued from previous page)

```
    }
}


?>
```

9.302.1 Suggestions

- Refactor code to avoid nesting
- Export some nested blocks to an external method or function

Name	Default	Type	Description
maxLevel	4	integer	Maximum level of nesting for control flow structures in one scope.

Short name	Structures/MaxLevelOfIdentation
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.303 Mbstring Third Arg

Some mbstring functions use the third argument for offset, not for encoding.

Those are the following functions :

- mb_strrichr()
- mb_stripos()
- mb_strrpos()
- mb_strstr()
- mb_stristr()
- mb_strpos()
- mb_strripos()
- mb_strrchr()
- mb_strrichr()
- mb_substr()

```
<?php

// Display BC
echo mb_substr('ABC', 1 , 2, 'UTF8');

// Yields Warning: mb_substr() expects parameter 3 to be int, string given
// Display 0 (aka, substring from 0, for length (int) 'UTF8' => 0)
echo mb_substr('ABC', 1 ,'UTF8');
```

(continues on next page)

(continued from previous page)

```

?>
```

See also mb_substr() manual pages.

9.303.1 Suggestions

- Add a third argument
- Use the default encoding (aka, omit both third AND fourth argument)

Short name	Structures/MbstringThirdArg
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.304 Mbstring Unknown Encoding

The encoding used is not known to the ext/mbstring extension.

This analysis takes in charge all `mbstring` encoding and aliases. The full list of supported mbstring encoding is available with mb_list_encodings(). Each encoding alias is available with mb_encoding_aliases().

```php
<?php

// Invalid encoding
$str = mb_strtolower($str, 'utf_8');

// Valid encoding
$str = mb_strtolower($str, 'utf8');
$str = mb_strtolower($str, 'UTF8');
$str = mb_strtolower($str, 'UTF-8');

?>
```

See also ext/mbstring.

9.304.1 Suggestions

- Use a valid mbstring encoding

Short name	Structures/MbstringUnknownEncoding
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.305 Memoize MagicCall

Cache calls to magic methods in local variable. Local cache is faster than calling again the magic method as soon as the second call, provided that the value hasn't changed.

__get is slower, as it turns a simple member access into a full method call.

```php
<?php

class x {
    private $values = array();

    function __get($name) {
        return $this->values[$name];
    }
    // more code to set values to this class
}

function foo(x $b) {
    $a = $b->a;
    $c = $b->c;

    $d = $c;     // using local cache, no new access to $b->__get($name)
    $e = $b->a;  // Second access to $b->a, through __get
}

function bar(x $b) {
    $a = $b->a;
    $c = $b->c;

    $b->bar2(); // this changes $b->a and $b->c, but we don't see it

    $d = $b->c;
    $e = $b->a;  // Second access to $b->a, through __get
}

?>
```

The caching is not possible if the processing of the object changes the value of the property.

See also __get performance questions with PHP, *Make Magic Concrete* and Benchmarking magic.

9.305.1 Suggestions

- Cache the value in a local variable, and reuse that variable
- Make the property concrete in the class, so as to avoid __get() altogether

Name	Default	Type	Description
minMagicCallsToGet	2	integer	Minimal number of calls of a magic property to make it worth locally caching.

Short name	Performances/MemoizeMagicCall
Rulesets	*Analyze*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.306 Merge If Then

Two successive if/then into one, by merging the two conditions.

```
<?php

// two merge conditions
if ($a == 1 && $b == 2) {
    // doSomething()
}

// two distinct conditions
// two nesting
if ($a == 1) {
    if ($b == 2) {
        // doSomething()
    }
}

?>
```

9.306.1 Suggestions

- Merge the two structures into one

Short name	Structures/MergeIfThen
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.307 Method Collision Traits

Two or more traits are included in the same class, and they have methods collisions.

Those collisions should be solved with a `use` expression. When they are not, PHP stops execution with a fatal error : `Trait method M has not been applied, because there are collisions with other trait methods on C.`

```
<?php

trait A {
    public function A() {}
    public function M() {}
}
```

(continues on next page)

(continued from previous page)

```

trait B {
    public function B() {}
    public function M() {}
}

class C {
    use  A, B;
}

class D {
    use  A, B{
        B::M insteadof A;
    };
}

?>
```

The code above lints, but doesn't execute.

See also Traits.

Short name	Traits/MethodCollisionTraits
Rulesets	*Analyze*, *LintButWontExec*
Severity	Critical
Time To Fix	Quick (30 mins)

9.308 Method Could Be Private Method

The following methods are never used outside their class of definition. Given the analyzed code, they could be set as private.

```
<?php

class foo {
    public function couldBePrivate() {}
    public function cantdBePrivate() {}

    function bar() {
        // couldBePrivate is used internally.
        $this->couldBePrivate();
    }
}

class foo2 extends foo {
    function bar2() {
        // cantdBePrivate is used in a child class.
        $this->cantdBePrivate();
    }
}

//couldBePrivate() is not used outside
$foo = new foo();
```

(continues on next page)

(continued from previous page)

```
//cantdBePrivate is used outside the class
$foo->cantdBePrivate();

?>
```

Note that dynamic properties (such as $x->$y) are not taken into account.

Short name	Classes/CouldBePrivateMethod
Rulesets	*ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.309 Method Could Be Static

A method that doesn't make any usage of $this could be turned into a static method.

While static methods are usually harder to handle, recognizing the static status is a first step before turning the method into a standalone function.

```
<?php

class foo {
    static $property = 1;

    // legit static method
    static function staticMethod() {
        return self::$property;
    }

    // This is not using $this, and could be static
    function nonStaticMethod() {
        return self::$property;
    }

    // This is not using $this nor self, could be a standalone function
    function nonStaticMethod() {
        return self::$property;
    }
}

?>
```

9.309.1 Suggestions

- Make the method static
- Make the method a standalone function
- Make use of $this in the method : may be it was forgotten.

Short name	Classes/CouldBeStatic
Rulesets	none
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*FuelCMS*, *ExpressionEngine*

9.310 Method Signature Must Be Compatible

Make sure methods signature are compatible.

PHP generates the infamous Fatal error at execution : `Declaration of FooParent\:\:Bar() must be compatible with FooChildren\:\:Bar()`

```php
<?php

class x {
    function xa() {}
}

class xxx extends xx {
    function xa($a) {}
}

?>
```

9.310.1 Suggestions

- Fix the child class method() signature.
- Fix the parent class method() signature, after checking that it won't affect the other children.

Short name	Classes/MethodSignatureMustBeCompatible
Rulesets	*Analyze*, *LintButWontExec*
Severity	Critical
Time To Fix	Quick (30 mins)

9.311 Methodcall On New

It is possible to call a method right at object instantiation.

This syntax was added in PHP 5.4+. Before, this was not possible : the object had to be stored in a variable first.

```php
<?php

// Data is collected
$data = data_source();

// Data is saved, but won't be reused from this databaseRow object. It may be ignored.
$result = (new databaseRow($data))->save();

```

(continues on next page)

(continued from previous page)

```
// The actual result of the save() is collected and tested.
if ($result !== true) {
    processSaveError($data);
}

?>
```

This syntax is interesting when the object is not reused, and may be discarded

Short name	Php/MethodCallOnNew
Rulesets	*CompatibilityPHP53*
Php Version	With PHP 5.4 and more recent
Severity	Major
Time To Fix	Quick (30 mins)

9.312 Methods Without Return

List of all the function, closures, methods that have no explicit return.

Functions that hold the `void` return type are omitted.

```
<?php

// With return null : Explicitly not returning
function withExplicitReturn($a = 1) {
    $a++;
    return null;
}

// Without indication
function withoutExplicitReturn($a = 1) {
    $a++;
}

// With return type void : Explicitly not returning
function withExplicitReturnType($a = 1) : void {
    $a++;
}

?>
```

See also return.

9.312.1 Suggestions

- Add the returntype 'void' to make this explicit behavior

Short name	Functions/WithoutReturn
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

9.313 Minus One On Error

Some PHP native functions return -1 on error. They also return 1 in case of success, and 0 in case of failure. This leads to confusions.

In case the native function is used as a condition without explicit comparison, PHP type cast the return value to a boolean. In this case, -1 and 1 are both converted to true, and the condition applies. This means that an error situation is mistaken for a successful event.

```
<?php

// Proper check of the return value
if (openssl_verify($data, $signature, $public) === 1) {
    $this->loginAsUser($user);
}

// if this call fails, it returns -1, and is confused with true
if (openssl_verify($data, $signature, $public)) {
    $this->loginAsUser($user);
}
?>
```

This analysis searches for if/then structures, ternary operators inside while() / do…'while() <https://www.php.net/manual/en/control-structures.while.php>'_ loops.

See also Can you spot the vulnerability? (openssl_verify) and Incorrect Signature Verification.

9.313.1 Suggestions

- Compare explicitly the return value to 1

Short name	Security/MinusOneOnError
Rulesets	*Security*
Severity	Critical
Time To Fix	Instant (5 mins)

9.314 Mismatch Parameter And Type

When the name of the parameter contradicts the type of the parameter.

This is mostly semantics, so it will affect the coder and the auditor of the code. PHP is immune to those errors.

```
<?php

// There is a discrepancy between the typehint and the name of the variable
function foo(int $string) { }

// The parameter name is practising coding convention typehints
function bar(int $int) { }

?>
```

9.314.1 Suggestions

- Synch the name of the parameter and the typehint.

Short name	Functions/MismatchParameterAndType
Rulesets	*Analyze*, *Semantics*
Severity	Minor
Time To Fix	Quick (30 mins)

9.315 Mismatch Parameter Name

Parameter name change in overwritten method. This may lead to errors when using PHP 8.0 named arguments.

PHP use the name of the parameter in the method whose code is executed. When the name change between the method and the overwritten method, the consistency is broken.

```
<?php

class x {
    function getValue($name) {}
}

class y extends x {
    // consistent with the method above
    function getValue($name) {}
}

class z extends x {
    // inconsistent with the method above
    function getValue($label) {}
}

?>
```

9.315.1 Suggestions

- Make sure all the names are the same, between methods

Short name	Functions/MismatchParameterName
Rulesets	*Analyze*, *CompatibilityPHP80*
Severity	Minor
Time To Fix	Quick (30 mins)

9.316 Mismatch Properties Typehints

Properties must match within the same family.

When a property is declared both in a parent class, and a child class, they must have the same type. The same type includes a possible null value.

This doesn't apply to private properties, which are only visible locally.

```
<?php

// property $p is declared as an object of type a class x {

    protected A $p;

}

// property $p is declared again, this time without a type class a extends x {

    protected $p;

}
```

This code will lint, but not execute.

9.316.1 Suggestions

- Remove some of the property declarations, and only keep it in the highest ranking parent
- Match the typehints of the property declarations
- Make the properties private
- Remove the child class (or the parent class)

Short name	Classes/MismatchProperties
Rulesets	*Analyze*, *LintButWontExec*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.317 Mismatch Type And Default

The argument typehint and its default value don’t match.

The code may lint and load, and even work when the arguments are provided. Though, PHP won’t eventually execute it.

Most of the mismatch problems are caught by PHP at linting time. It displays the following error message : ‘Argument 1 passed to foo() must be of the type integer, string given’.

The default value may be a constant (normal or class constant) : as such, PHP might find its value only at execution time, from another include. As such, PHP doesn’t report anything about the situation at compile time.

The default value may also be a constant scalar expression : since PHP 7, some of the simple operators such as +, -, , %, ‘* <https://www.php.net/manual/en/language.operators.arithmetic.php>‘_, etc. are available to build default values. Among them, the ternary operator and Coalesce. Again, those expression may be only evaluated at execution time, when the value of the constants are known.

```
<?php

// bad definition : the string is actually an integer
const STRING = 3;

function foo(string $s = STRING) {
    echo $s;
}

```

(continues on next page)

(continued from previous page)

```
// works without problem
foo('string');

// Fatal error at compile time
foo();

// Fail only at execution time (missing D), and when default is needed
function foo2(string $s = D ? null : array()) {
    echo $s;
}

?>
```

PHP reports typehint and default mismatch at compilation time, unless there is a static expression that can't be resolved within the compiled file : then it is checked only at runtime, leading to a Fatal error.

See also Type declarations.

9.317.1 Suggestions

- Match the typehint with the default value
- Do not rely on PHP type juggling to change the type on the fly

Short name	Functions/MismatchTypeAndDefault
Rulesets	*Analyze*, *LintButWontExec*, *Typechecks*
Severity	Critical
Time To Fix	Slow (1 hour)

9.318 Mismatched Default Arguments

Arguments are relayed from one method to the other, and the arguments have different default values.

Although it is possible to have different default values, it is worth checking why this is actually the case.

```
<?php

function foo($a = null, $b = array() ) {
    // foo method calls directly bar.
    // When argument are provided, it's OK
    // When argument are omited, the default value is not the same as the next method
    bar($a, $b);
}

function bar($c = 1, $d = array() ) {

}

?>
```

9.318.1 Suggestions

- Synchronize default values to avoid surprises
- Drop some of the default values

Short name	Functions/MismatchedDefaultArguments
Rulesets	*Analyze*, *Typechecks*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*SPIP*

9.319 Mismatched Ternary Alternatives

A ternary operator should yield the same type on both branches.

Ternary operator applies a condition, and yield two different results. Those results will then be processed by code that expects the same types. It is recommended to match the types on both branches of the ternary operator.

```
<?php

// $object may end up in a very unstable state
$object = ($type == 'Type') ? new $type() : null;

//same result are provided by both alternative, though process is very different
$result = ($type == 'Addition') ? $a + $b : $a * $b;

//Currently, this is omitted
$a = 1;
$result = empty($condition) ? $a : 'default value';
$result = empty($condition) ? $a : getDefaultValue();

?>
```

9.319.1 Suggestions

- Use compatible data type in both branch of the alternative
- Turn the ternary into a if/then, with different processing

Short name	Structures/MismatchedTernary
Rulesets	*Analyze*, *Suggestions*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*phpadsnew*, *OpenEMR*

9.320 Mismatched Typehint

Relayed arguments don’t have the same typehint.

Typehint acts as a filter method. When an object is checked with a first class, and then checked again with a second distinct class, the whole process is always false : $a can't be of two different classes at the same time.

```
<?php

// Foo() calls bar()
function foo(A $a, B $b) {
    bar($a, $b);
}

// $a is of A typehint in both methods, but
// $b is of B then BB typehing
function bar(A $a, BB $b) {

}

?>
```

Note : This analysis currently doesn't check generalisation of classes : for example, when B is a child of BB, it is still reported as a mismatch.

9.320.1 Suggestions

- Ensure that the default value match the expected typehint.

Short name	Functions/MismatchedTypehint
Rulesets	*Analyze*, *Typechecks*
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*WordPress*

9.321 Missing Abstract Method

Abstract methods must have a non-abstract version for the class to be complete. A class that is missing one abstract definition cannot be instantiated.

```
<?php

// This is a valid definition
class b extends a {
    function foo() {}
    function bar() {}
}

// This compiles, but will emit a fatal error if instantiated
class c extends a {
    function bar() {}
}

// This illustration lint but doesn't run.
// moving this class at the beginning of the code will make lint fail
abstract class a {
```

(continues on next page)

(continued from previous page)

```
    abstract function foo() ;
}

?>
```

See also Classes Abstraction.

9.321.1 Suggestions

- Implement the missing methods
- Remove the partially implemented class
- Mark the partially implemented class abstract

Short name	Classes/MissingAbstractMethod
Rulesets	*Analyze*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.322 Missing Cases In Switch

It seems that some cases are missing in this switch structure.

When comparing two different switch() structures, it appears that some cases are missing in one of them. The set of cases are almost identical, but one of the values are missing.

Switch() structures using strings as literals are compared in this analysis. When the discrepancy between two lists is below 25%, both switches are reported.

```
<?php

// This switch operates on a, b, c, d and default
switch($a) {
    case 'a': doSomethingA(); break 1;
    case 'b': doSomethingB(); break 1;
    case 'c': doSomethingC(); break 1;
    case 'd': doSomethingD(); break 1;
    default: doNothing();
}

// This switch operates on a, b, d and default
switch($o->p) {
    case 'a': doSomethingA(); break 1;
    case 'b': doSomethingB(); break 1;

    case 'd': doSomethingD(); break 1;
    default: doNothing();
}

?>
```

In the example, one may argue that the 'c' case is actually handled by the 'default' case. Otherwise, business logic may request that omission.

9.322.1 Suggestions

- Add the missing cases
- Add comments to mention that missing cases are processed in the default case

Short name	Structures/MissingCases
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*Tikiwiki*

9.323 Missing Include

The included files doesn’t exists in the repository. The inclusions target a files that doesn’t exist.

The analysis works with every type of inclusion : include(), require(), include_once() and require_once(). It also works with parenthesis when used as parameter delimiter.

The analysis doesn’t take into account `include_path`. This may yield false positives.

```php
<?php

include 'non_existent.php';

// variables are not resolved. This won't be reported.
require ($path.'non_existent.php');

?>
```

Missing included files may lead to a fatal error, a warning or other error later in the execution.

Name	Default	Type	Description
constant_or_variable_name	100	string	Literal value to be used when including files. For example, by configuring ‘Files_MissingInclude[“HOME_DIR”] = “/tmp/myDir/”;’, then ‘include HOME_DIR . “my_class.php”; will be actually be used as ‘/tmp/myDir/my_class.php’. Constants must be configured with their correct case. Variable must be configured with their initial ‘$’. Configure any number of variable and constant names.

Short name	Files/MissingInclude
Rulesets	*Analyze*
Severity	Critical
Time To Fix	Instant (5 mins)

9.324 Missing New ?

This functioncall looks like a class instantiation that is missing the new keyword.

Any function definition was found for that function, but a class with that name was. New is probably missing.

```
<?php

// Functioncall
$a = foo();

// Class definition
class foo {}
// Function definition
function foo {}


// Functioncall
$a = BAR;

// Function definition
class bar {}
// Constant definition
const BAR = 1;


?>
```

9.324.1 Suggestions

- Add the new
- Rename the class to distinguish it from the function
- Rename the function to distinguish it from the class

Short name	Structures/MissingNew
Rulesets	*Analyze*
Severity	Critical
Time To Fix	Instant (5 mins)

9.325 Missing Parenthesis

Add parenthesis to those expression to prevent bugs.

```
<?php

// Missing some parenthesis!!
if (!$a instanceof Stdclass) {
    print Not\n;
} else {
    print Is\n;
}

// Could this addition be actually
$c = -$a + $b;

// This one ?
$c = -($a + $b);
```

(continues on next page)

(continued from previous page)

```

?>
```

See also Operators Precedence.

Short name	Structures/MissingParenthesis
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)

9.326 Missing Returntype In Method

The specified typehints are not sufficient. The code of the method may return other types, which are not specified.

```
<?php

function fooSN() : ?string  {
    return shell_exec('ls -hla');
}

// shell_exec() may return null or string. Here, only string in specified for fooS,␣
↪and that may lead to a Fatal error
function fooS() : string  {
    return shell_exec('ls -hla');
}

function bar() : int {
    return rand(0, 10) ? 1 : b;
}

?>
```

The analysis reports a method when it finds other return types than the one expected. In the case of multiple typehints, as for the last example, the PHP code may require an upgrade to PHP 8.0.

9.326.1 Suggestions

- Update the typehint to accept more types
- Update the code of the method to fit the expected returntype

Short name	Typehints/MissingReturntype
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

9.327 Missing Typehint

No typehint was found for this parameter, or as a return type for the function.

void is considered a specified typehint, and is not reported here.

```
<?php

function foo($no_typehint) : void {}

function no_return_type() {}

?>
```

See also Type Declaration.

9.327.1 Suggestions

-

Short name	Functions/MissingTypehint
Rulesets	*Typechecks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.328 Mistaken Concatenation

A unexpected structure is built for initialization. It may be a typo that creates an unwanted expression.

```
<?php

// This 'cd' is unexpected. Isn't it 'c', 'd' ?
$array = array('a', 'b', 'c'. 'd');
$array = array('a', 'b', 'c', 'd');

// This 4.5 is unexpected. Isn't it 4, 5 ?
$array = array(1, 2, 3, 4.5);
$array = array(1, 2, 3, 4, 5);

?>
```

Short name	Arrays/MistakenConcatenation
Rulesets	*Coding Conventions*
Severity	Major
Time To Fix	Instant (5 mins)

9.329 Mixed Concat And Interpolation

Mixed usage of concatenation and string interpolation is error prone. It is harder to read, and leads to overlooking the concatenation or the interpolation.

```
<?php

```

(continues on next page)

(continued from previous page)

```
// Concatenation string
$a = $b . 'c' . $d;

// Interpolation strings
$a = {$b}c{$d};    // regular form
$a = {$b}c$d;      // irregular form

// Mixed Concatenation and Interpolation string
$a = {$b}c . $d;
$a = $b . c$d;
$a = $b . c{$d};

// Mixed Concatenation and Interpolation string with constant
$a = {$b}c . CONSTANT;

?>
```

Fixing this issue has no impact on the output. It makes code less error prone.

There are some situations where using concatenation are compulsory : when using a constant, calling a function, running a complex expression or make use of the escape sequence. You may also consider pushing the storing of such expression in a local variable.

9.329.1 Suggestions

- Only use one type of variable usage : either interpolation, or concatenation

Short name	Structures/MixedConcatInterpolation
Rulesets	*Coding Conventions*, *Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*SuiteCrm*, *Edusoho*

9.330 Mixed Keys Arrays

Avoid mixing constants and literals in array keys.

When defining default values in arrays, it is recommended to avoid mixing constants and literals, as PHP may mistake them and overwrite the previous with the latter.

Either switch to a newer version of PHP (5.5 or newer), or make sure the resulting array hold the expected data. If not, reorder the definitions.

```
<?php

const ONE = 1;

$a = [ 1   => 2,
       ONE => 3];

?>
```

9.330.1 Suggestions

- Use only literals or constants when building the array

Short name	Arrays/MixedKeys
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*
Php Version	With PHP 5.6 and more recent
Severity	Minor
Time To Fix	Slow (1 hour)

9.331 Mkdir Default

mkdir() gives universal access to created folders, by default. It is recommended to gives limited set of rights (0755, 0700), or to explicitly set the rights to 0777.

```
<?php

// By default, this dir is 777
mkdir('/path/to/dir');

// Explicitely, this is wanted. It may also be audited easily
mkdir('/path/to/dir', 0777);

// This dir is limited to the current user.
mkdir('/path/to/dir', 0700);

?>
```

See also Why 777 Folder Permissions are a Security Risk.

9.331.1 Suggestions

- Always use the lowest possible privileges on folders
- Don’t use the PHP default : at least, make it explicit that the ‘universal’ rights are voluntary

Short name	Security/MkdirDefault
Rulesets	*Security*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Mautic*, *OpenEMR*

9.332 Modernize Empty With Expression

empty() accepts expressions as argument. This feature was added in PHP 5.5.

There is no need to store the expression in a variable before testing, unless it is reused later.

```
<?php

// PHP 5.5+ empty() usage
if (empty(foo($b . $c))) {
    doSomethingWithoutA();
}

// Compatible empty() usage
$a = foo($b . $c);
if (empty($a)) {
    doSomethingWithoutA();
}

// $a2 is reused, storage is legit
$a2 = strtolower($b . $c);
if (empty($a2)) {
    doSomething();
} else {
    echo $a2;
}

?>
```

See also empty() and empty() supports arbitrary expressions.

9.332.1 Suggestions

- Avoid the temporary variable, and use directly empty()

Short name	Structures/ModernEmpty
Rulesets	*Analyze*
Php Version	With PHP 5.5 and more recent
Severity	Minor
Time To Fix	Quick (30 mins)

9.333 Modified Typed Parameter

Reports modified parameters, which have a non-scalar typehint. Such variables should not be changed within the body of the method. Unlike typed properties, which always hold the expected type, typed parameters are only garanteed type at the beginning of the method block.

```
<?php

class x {

    function foo(Y $y) {
        // $y is type Y

        // A cast version of $y is stored into $yAsString. $y is untouched.
        $yAsString = (string) $y;

        // $y is of type 'int', now.
```

(continues on next page)

(continued from previous page)

```
        $y = 1;

        // Some more code

        // display the string version.
        echo $yAsString;
        // so, Y $y is now raising an error
        echo $y->name;
    }
}

?>
```

This problem doesn't apply to scalar types : by default, PHP pass scalar parameters by value, not by reference. Class types are always passed by reference.

This problem is similar to **'Classes/DontUnsetProperties'_** : the static specification of the property may be unset, leading to confusing 'undefined property', while the class hold the property definition.

9.333.1 Suggestions

- Use different variable names when convertir a parameter to a different type.
- Only use methods and properties calls on a typed parameter.

Short name	Functions/ModifyTypedParameter
Rulesets	*Analyze*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.334 Multiple Alias Definitions

Some aliases are representing different classes across the repository. This leads to potential confusion.

Across an application, it is recommended to use the same namespace for one alias. Failing to do this lead to the same keyword to represent different values in different files, with different behavior. Those are hard to find bugs.

```
<?php

namespace A {
    use d\d; // aka D
}

// Those are usually in different files, rather than just different namespaces.

namespace B {
    use b\c as D; // also D. This could be named something else
}

?>
```

9.334.1 Suggestions

- Give more specific names to classes
- Use an alias 'use AB ac BC' to give locally another name

Short name	Namespaces/MultipleAliasDefinitions
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*ChurchCRM*, *Phinx*

9.335 Multiple Alias Definitions Per File

Avoid aliasing the same name with different aliases. This leads to confusion.

```
<?php

// first occurrence
use name\space\ClasseName;

// when this happens, several other uses are mentionned

// name\space\ClasseName has now two names
use name\space\ClasseName as anotherName;

?>
```

See also Namespaces/MultipleAliasDefinition.

Short name	Namespaces/MultipleAliasDefinitionPerFile
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Slow (1 hour)

9.336 Multiple Class Declarations

It is possible to declare several times the same class in the code. PHP will not mention it until execution time, since declarations may be conditional.

```
<?php

$a = 1;

// Conditional declaration
if ($a == 1) {
    class foo {
        function method() { echo 'class 1';}
    }
} else {
    class foo {
```

(continues on next page)

(continued from previous page)

```
        function method() { echo 'class 2';}
    }
}

(new foo())->method();
?>
```

It is recommended to avoid declaring several times the same class in the code. The best practice is to separate them with namespaces, they are for here for that purpose. In case those two classes are to be used interchangeably, the best is to use an abstract class or an interface.

9.336.1 Suggestions

- Store classes with different names in different namespaces
- Change the name of the classes and give them a common interface to allow from common behavior

Short name	Classes/MultipleDeclarations
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)

9.337 Multiple Classes In One File

It is regarded as a bad practice to store several classes in the same file. This is usually done to make life of __autoload() easier.

It is often unexpected to find class `foo` in the `bar.php` file. This is also the case for interfaces and traits.

```
<?php

// three classes in the same file
class foo {}
class bar {}
class foobar{}

?>
```

One good reason to have multiple classes in one file is to reduce include time by providing everything into one nice include.

See also Is it a bad practice to have multiple classes in the same file?.

9.337.1 Suggestions

- Split the file into smaller files, one for each class

Short name	Classes/MultipleClassesInFile
Rulesets	*Coding Conventions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.338 Multiple Constant Definition

Some constants are defined several times in your code. This will lead to a fatal error, if they are defined during the same execution.

Multiple definitions may happens at bootstrap, when the application code is collecting information about the current environment. It may also happen at inclusion time, which one set of constant being loaded, while other definition are not, avoiding conflict. Both are false positive.

```
<?php

// OS is defined twice.
if (PHP_OS == 'Windows') {
    define('OS', 'Win');
} else {
    define('OS', 'Other');
}

?>
```

9.338.1 Suggestions

- Move the constants to a class, and include the right class based on control flow.
- Give different names to the constants, and keep the condition close to utilisation.
- Move the constants to an external configuration file : it will be easier to identify that those constants may change.

Short name	Constants/MultipleConstantDefinition
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Dolibarr*, *OpenConf*

9.339 Multiple Declaration Of Strict_types

At least two declare() commands are declaring *strict_types* in one file. Only one is sufficient, and should be the first expression in the file.

Indeed, any *strict_types* set to 1 will have the final word. Setting *strict_types* to 0 will not revert the configuration, wherever is this call made.

```
<?php
declare(strict_types=1);
declare(strict_types=1);

// rest of the code

?>
```

See also Declare.

9.339.1 Suggestions

- Just remove all but one of them.

Short name	Php/MultipleDeclareStrict
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.340 Multiple Definition Of The Same Argument

A method's signature is holding twice (or more) the same argument. For example, function x ($a, $a) { `...` }.

This is accepted as is by PHP 5, and the last parameter's value will be assigned to the variable. PHP 7.0 and more recent has dropped this feature, and reports a fatal error when linting the code.

```
<?php
  function x ($a, $a) { print $a; };
  x(1,2); => display 2

    // special case with a closure :
  function ($a) use ($a) { print $a; };
  x(1,2); => display 2

?>
```

However, this is not common programming practise : all arguments should be named differently.

See also Prepare for PHP 7 error messages (part 3).

9.340.1 Suggestions

- Give different names to different parameters

Short name	Functions/MultipleSameArguments
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Instant (5 mins)
ClearPHP	all-unique-arguments

9.341 Multiple Exceptions Catch()

It is possible to have several distinct exceptions class caught by the same catch, preventing code repetition.

This is a new feature since PHP 7.1.

```
<?php

// PHP 7.1 and more recent
```

(continues on next page)

(continued from previous page)

```
try {
    throw new someException();
} catch (Single $s) {
    doSomething();
} catch (oneType | anotherType $s) {
    processIdentically();
} finally {

}

// PHP 7.0 and older
try {
    throw new someException();
} catch (Single $s) {
    doSomething();
} catch (oneType $s) {
    processIdentically();
} catch (anotherType $s) {
    processIdentically();
} finally {

}

?>
```

This is a backward incompatible feature of PHP 7.1.

Short name	Exceptions/MultipleCatch
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Severity	Major
Time To Fix	Quick (30 mins)

9.342 Multiple Identical Trait Or Interface

There is no need to use the same trait, or implements the same interface more than once.

Up to PHP 7.1 (at least), this doesn't raise any warning. Traits are only imported once, and interfaces may be implemented as many times as wanted.

```
<?php

class foo {
    use t3,t3,t3;
}

class bar implements i,i,i {

}

?>
```

9.342.1 Suggestions

- Remove the duplicate trait or interfaces

Short name	Classes/MultipleTraitOrInterface
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)

9.343 Multiple Index Definition

Indexes that are defined multiple times in the same array.

```
<?php
    // Multiple identical keys
    $x = array(1 => 2,
               2 => 3,
               1 => 3);

    // Multiple identical keys (sneaky version)
    $x = array(1 => 2,
               1.1 => 3,
               true => 4);

    // Multiple identical keys (automated version)
    $x = array(1 => 2,
               3,         // This will be index 2
               2 => 4);   // this index is overwritten
?>
```

They are indeed overwriting each other. This is most probably a typo.

9.343.1 Suggestions

- Review your code and check that arrays only have keys defined once.
- Review carefully your code and check indirect values, like constants, static constants.

Short name	Arrays/MultipleIdenticalKeys
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Magento*, *MediaWiki*

9.344 Multiple Type Variable

Avoid using the same variable with different types of data.

It is recommended to use different names for differently typed data, while processing them. This prevents errors where one believe the variable holds the former type, while it has already been cast to the later.

Incrementing variables, with math operations or concatenation, is OK : the content changes, but not the type. And casting the variable without storing it in itself is OK.

```php
<?php

// $x is an array
$x = range('a', 'z');
// $x is now a string
$x = join('', $x);
$c = count($x); // $x is not an array anymore


// $letters is an array
$letters = range('a', 'z');
// $alphabet is a string
$alphabet = join('', $letters);

// Here, $letters is cast by PHP, but the variable is changed.
if ($letters) {
    $count = count($letters); // $letters is still an array
}

?>
```

9.344.1 Suggestions

- Use a class that accepts one type of argument, and exports another type of argument.
- Use different variable for each type of data format : $rows (for array), $list (for implode(‘’, $rows))
- Pass the final result as argument to another method, avoiding the temporary variable

Short name	Structures/MultipleTypeVariable
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Typo3*, *Vanilla*

9.345 Multiple Unset()

Unset() accepts multiple arguments, unsetting them one after each other. It is more efficient to call unset() once, than multiple times.

```php
<?php

// One call to unset only
unset($a, $b, $c, $d);

// Too many calls to unset
unset($a);
unset($b);
unset($c);
unset($d);
```

(continues on next page)

(continued from previous page)

```
?>
```

See also unset.

9.345.1 Suggestions

- Merge all unset into one call

Short name	Structures/MultipleUnset
Rulesets	*Suggestions*, *php-cs-fixable*
Severity	Minor
Time To Fix	Quick (30 mins)

9.346 Multiple Usage Of Same Trait

The same trait is used several times. One trait usage is sufficient.

```
<?php

// C is used twice, and could be dropped from B
trait A { use B, C;}
trait B { use C;}

?>
```

PHP doesn't raise any error when traits are included multiple times.

See also Traits.

9.346.1 Suggestions

- Remove any multiple traits from use expressions
- Review the class tree, and remove any trait mentioned multiple times

Short name	Traits/MultipleUsage
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*NextCloud*

9.347 Multiples Identical Case

Some cases are defined multiple times, but only one will be processed. Check the list of cases, and remove the extra one.

Exakat tries to find the value of the case as much as possible, and ignore any dynamic cases (using variables).

```
<?php

const A = 1;

case ($x) {
    case 1 :
        break;
    case true:    // This is a duplicate of the previous
        break;
    case 1 + 0:   // This is a duplicate of the previous
        break;
    case 1.0 :    // This is a duplicate of the previous
        break;
    case A :      // The A constant is actually 1
        break;
    case $y  :    // This is not reported.
        break;
    default:

}
?>
```

9.347.1 Suggestions

- Remove the double case
- Change the case to another and rightful value

Short name	Structures/MultipleDefinedCase
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
ClearPHP	no-duplicate-case
Examples	*SugarCrm*, *ExpressionEngine*

9.348 Multiply By One

Multiplying by 1 is a fancy type cast.

If it is used to type cast a value to number, then casting (integer) or (real) is clearer. This behavior may change with PHP 7.1, which has unified the behavior of all hidden casts.

```
<?php

// Still the same value than $m, but now cast to integer or real
$m = $m * 1;

// Still the same value than $m, but now cast to integer or real
$n *= 1;

// make typecasting clear, and merge it with the producing call.
$n = (int) $n;
```

(continues on next page)

(continued from previous page)

```

?>
```

See also Type Juggling

9.348.1 Suggestions

- Typecast to (int) or (float) for better readability
- Skip useless math operation altogether

Short name	Structures/MultiplyByOne
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
ClearPHP	no-useless-math
Examples	*SugarCrm*, *Edusoho*

9.349 Must Call Parent Constructor

Some PHP native classes require a call to parent::__construct() to be stable.

As of PHP 7.3, two classes currently need that call : SplTempFileObject and SplFileObject.

The error is only emitted if the class is instantiated, and a parent class is called.

```
<?php

class mySplFileObject extends \SplFileObject {
    public function __construct()    {
        // Forgottent call to parent::__construct()
    }
}

(new mySplFileObject())->passthru();
?>
```

See also Why, php? WHY???.

9.349.1 Suggestions

- Add a call to the parent's constructor
- Remove the extension of the parent class

Short name	Php/MustCallParentConstructor
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)

9.350 Must Return Methods

The following methods are expected to return a value that will be used later. Without return, they are useless.

Methods that must return are : __get(), __isset(), __sleep(), __toString(), __set_state(), __invoke(), __debugInfo(). Methods that may not return, but are often expected to : __call(), __callStatic().

```
<?php

class foo {
    public function __isset($a) {
        // returning something useful
        return isset($this->$var[$a]);
    }

    public function __get($a) {
        $this->$a++;
        // not returning...
    }

    public function __call($name, $args) {
        $this->$name(...$args);
        // not returning anything, but that's OK
    }

}
?>
```

9.350.1 Suggestions

- Add a return expression, with a valid data type
- Remove the return typehint

Short name	Functions/MustReturn
Rulesets	*Analyze*, *LintButWontExec*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)

9.351 Named Regex

Captured subpatterns may be named, for easier reference.

From the manual : It is possible to name a subpattern using the syntax `(?P<name>pattern)`. This subpattern will then be indexed in the matches array by its normal numeric position and also by name. PHP 5.2.2 introduced two alternative syntaxes `(?<name>pattern)` and `(?'name'pattern)`.

Naming subpatterns makes it easier to know what is read from the results of the subpattern : for example, `$r['name']` has more meaning than `$r[1]`.

Named subpatterns may also be shifted in the regex without impact on the resulting array.

```
<?php

$x = 'abc';
preg_match_all('/(?<name>a)/', $x, $r);
print_r($r[1]);
print_r($r['name']);

preg_match("/(?<name>a)(?'sub'b)/", $x, $s);
print $s[2];
print $s['sub'];

?>
```

See also Subpatterns.

9.351.1 Suggestions

- Use named regex, and stop using integer-named subpatterns

Short name	Structures/NamedRegex
Rulesets	*Suggestions*
Examples	*Phinx*, *shopware*

9.352 Negative Power

The power operator ** has higher precedence than the sign operators + and -.

This means that -2 ** 2 == -4. It is in fact, -(2 ** 2).

When using negative power, it is clearer to add parenthesis or to use the pow() function, which has no such ambiguity :

```
<?php

// -2 to the power of 2 (a square)
pow(-2, 2) == 4;

// minus 2 to the power of 2 (a negative square)
-2 ** 2 == -(2 ** 2) == 4;

?>
```

9.352.1 Suggestions

- Avoid negative number, as operands of **
- Use parenthesis with negative numbers and **

Short name	Structures/NegativePow
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)

9.353 Negative Start Index In Array

Negative starting index in arrays changed in PHP 8.0. Until then, they were ignored, and automatic index started always at 0. Since PHP 8.0, the next index is calculated.

The behavior will break code that relies on automatic index in arrays, when a negative index is used for a starter.

```
<?php

$x = [-5 => 2];
$x[] = 3;

print_r($x);

/*
PHP 7.4 and older
Array
(
    [-5] => 2
    [0] => 3
)
*/

/*
PHP 8.0 and more recent
Array
(
    [-5] => 2
    [-4] => 3
)
*/

?>
```

See also PHP RFC: Arrays starting with a negative index.

9.353.1 Suggestions

- Explicitely create the index, instead of using the automatic indexing
- Add an explicit index of 0 in the initial array, to set the automatic process in the right track
- Avoid using specified index in array, conjointly with automatic indexing.

Short name	Arrays/NegativeStart
Rulesets	*CompatibilityPHP80*
Severity	Minor
Time To Fix	Quick (30 mins)

9.354 Nested Ifthen

Three levels of ifthen is too much. The method should be split into smaller functions.

```
<?php

function foo($a, $b) {
    if ($a == 1) {
        // Second level, possibly too much already
        if ($b == 2) {

        }
    }
}

function bar($a, $b, $c) {
    if ($a == 1) {
        // Second level.
        if ($b == 2) {
            // Third level level.
            if ($c == 3) {
                // Too much
            }
        }
    }
}

?>
```

Name	Default	Type	Description
nestedIfthen	3	integer	Maximal number of acceptable nesting of if-then structures

Short name	Structures/NestedIfthen
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*LiveZilla*, *MediaWiki*

9.355 Nested Ternary

Ternary operators should not be nested too deep.

They are a convenient instruction to apply some condition, and avoid a if() structure. It works best when it is simple, like in a one liner.

However, ternary operators tends to make the syntax very difficult to read when they are nested. It is then recommended to use an if() structure, and make the whole code readable.

```
<?php

// Simple ternary expression
echo ($a == 1 ? $b : $c) ;

// Nested ternary expressions
echo ($a === 1 ? $d === 2 ? $b : $d : $d === 3 ? $e : $c) ;
echo ($a === 1 ? $d === 2 ? $f ===4 ? $g : $h : $d : $d === 3 ? $e : $i === 5 ? $j :
↪$k) ;
```

(continues on next page)

(continued from previous page)

```
//Previous expressions, written as a if / Then expression
if ($a === 1) {
    if ($d === 2) {
        echo $b;
    } else {
        echo $d;
    }
} else {
    if ($d === 3) {
        echo $e;
    } else {
        echo $c;
    }
}

if ($a === 1) {
    if ($d === 2) {
        if ($f === 4) {
            echo $g;
        } else {
            echo $h;
        }
    } else {
        echo $d;
    }
} else {
    if ($d === 3) {
        echo $e;
    } else {
        if ($i === 5) {
            echo $j;
        } else {
            echo $k;
        }
    }
}

?>
```

See also Nested Ternaries are Great.

9.355.1 Suggestions

- Replace ternaries by if/then structures.
- Replace ternaries by a functioncall : this provides more readability, offset the actual code, and gives room for making it different.

Short name	Structures/NestedTernary
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	no-nested-ternary
Examples	*SPIP*, *Zencart*

9.356 Nested Ternary Without Parenthesis

It is not allowed to nest ternary operator within itself, without parenthesis. This has been implemented in PHP 7.4.

The reason behind this feature is to keep the code expressive. See the Warning message for more explanations

```
<?php

$a ? 1 : ($b ? 2 : 3);

// Still valid, as not ambiguous
$a ? $b ? 1 : 2 : 3;

// Produces a warning
//Unparenthesized `a ? b : c ? d : e` is deprecated. Use either `(a ? b : c) ? d : e`␣
↪or `a ? b : (c ? d : e)`
$a ? 1 : $b ? 2 : 3;

?>
```

See also PHP RFC: Deprecate left-associative ternary operator.

9.356.1 Suggestions

- Add parenthesis to nested ternary calls

Short name	Php/NestedTernaryWithoutParenthesis
Rulesets	*CompatibilityPHP74*
Php Version	7.4-
Severity	Minor
Time To Fix	Quick (30 mins)

9.357 Never Used Parameter

When a parameter is never used at calltime, it may be turned into a local variable.

It seems that the parameter was set up initially, but never found its practical usage. It is never mentioned, and always fall back on its default value.

Parameter without a default value are reported by PHP, and are usually always filled.

```
<?php

// $b may be turned into a local var, it is unused
function foo($a, $b = 1) {
    return $a + $b;
}

// whenever foo is called, the 2nd arg is not mentionned
foo($a);
foo(3);
foo('a');
foo($c);
```

(continues on next page)

(continued from previous page)

```
?>
```

9.357.1 Suggestions

- Drop the unused argument in the method definition
- Actually use the argument when calling the method
- Drop the default value, and check warnings that mention usage of this parameter

Short name	Functions/NeverUsedParameter
Rulesets	*Analyze*, *Suggestions*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*Piwigo*

9.358 Never Used Properties

Properties that are never used. They are defined in a class or a trait, but they never actually used.

Properties are considered used when they are used locally, in the same class as their definition, or in a parent class : a parent class is always included with the current class.

On the other hand, properties which are defined in a class, but only used in children classes is considered unused, since children may also avoid using it.

```
<?php

class foo {
    public $usedProperty = 1;

    // Never used anywhere
    public $unusedProperty = 2;

    function bar() {
        // Used internally
        ++$this->usedProperty;
    }
}

class foo2  extends foo {
    function bar2() {
        // Used in child class
        ++$this->usedProperty;
    }
}

// Used externally
++$this->usedProperty;

?>
```

9.358.1 Suggestions

- Drop unused properties
- Change the name of the unused properties
- Move the properties to children classes
- Find usage for unused properties

Short name	Classes/PropertyNeverUsed
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*WordPress*

9.359 New Constants In PHP 7.2

The following constants are now native in PHP 7.2. It is advised to avoid using such names for constant before moving to this new version.

- PHP_OS_FAMILY
- PHP_FLOAT_DIG
- PHP_FLOAT_EPSILON
- PHP_FLOAT_MAX
- PHP_FLOAT_MIN
- SQLITE3_DETERMINISTIC
- CURLSSLOPT_NO_REVOKE
- CURLOPT_DEFAULT_PROTOCOL
- CURLOPT_STREAM_WEIGHT
- CURLMOPT_PUSHFUNCTION
- CURL_PUSH_OK
- CURL_PUSH_DENY
- CURL_HTTP_VERSION_2TLS
- CURLOPT_TFTP_NO_OPTIONS
- CURL_HTTP_VERSION_2_PRIOR_KNOWLEDGE
- CURLOPT_CONNECT_TO
- CURLOPT_TCP_FASTOPEN
- DNS_CAA

See also New global constants in 7.2.

Short name	Php/Php72NewConstants
Rulesets	*CompatibilityPHP72*
Php Version	With PHP 7.2 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.360 New Constants In PHP 7.4

The following constants are now native in PHP 7.4. It is advised to avoid using such names for constant before moving to this new version.

- `MB_ONIGURUMA_VERSION`
- `SO_LABEL`
- `SO_PEERLABEL`
- `SO_LISTENQLIMIT`
- `SO_LISTENQLEN`
- `SO_USER_COOKIE`
- `PHP_WINDOWS_EVENT_CTRL_C`
- `PHP_WINDOWS_EVENT_CTRL_BREAK`
- `TIDY_TAG_ARTICLE`
- `TIDY_TAG_ASIDE`
- `TIDY_TAG_AUDIO`
- `TIDY_TAG_BDI`
- `TIDY_TAG_CANVAS`
- `TIDY_TAG_COMMAND`
- `TIDY_TAG_DATALIST`
- `TIDY_TAG_DETAILS`
- `TIDY_TAG_DIALOG`
- `TIDY_TAG_FIGCAPTION`
- `TIDY_TAG_FIGURE`
- `TIDY_TAG_FOOTER`
- `TIDY_TAG_HEADER`
- `TIDY_TAG_HGROUP`
- `TIDY_TAG_MAIN`
- `TIDY_TAG_MARK`
- `TIDY_TAG_MENUITEM`
- `TIDY_TAG_METER`
- `TIDY_TAG_NAV`

- `TIDY_TAG_OUTPUT`
- `TIDY_TAG_PROGRESS`
- `TIDY_TAG_SECTION`
- `TIDY_TAG_SOURCE`
- `TIDY_TAG_SUMMARY`
- `TIDY_TAG_TEMPLATE`
- `TIDY_TAG_TIME`
- `TIDY_TAG_TRACK`
- `TIDY_TAG_VIDEO`
- `STREAM_CRYPTO_METHOD_TLSv1_3_CLIENT`
- `STREAM_CRYPTO_METHOD_TLSv1_3_SERVER`
- `STREAM_CRYPTO_PROTO_TLSv1_3`
- `T_COALESCE_EQUAL`
- `T_FN`

See also New global constants in 7.4.

9.360.1 Suggestions

- Move the constants to a new namespace
- Remove the old constants
- Rename the old constants

Short name	Php/Php74NewConstants
Rulesets	*CompatibilityPHP74*
Php Version	With PHP 7.4 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.361 New Functions In PHP 5.4

PHP introduced new functions in PHP 5.4. If there are defined functions with such names, there will be a conflict when upgrading. It is advised to change those functions' name.

Short name	Php/Php54NewFunctions
Rulesets	*CompatibilityPHP53*
Php Version	With PHP 5.3 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.362 New Functions In PHP 5.5

PHP introduced new functions in PHP 5.5. If you have already defined functions with such names, you will get a conflict when trying to upgrade. It is advised to change those functions' name.

Short name	Php/Php55NewFunctions
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*
Php Version	With PHP 5.5 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.363 New Functions In PHP 5.6

PHP introduced new functions in PHP 5.6. If you have already defined functions with such names, you will get a conflict when trying to upgrade. It is advised to change those functions' name.

Short name	Php/Php56NewFunctions
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Php Version	With PHP 5.6 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.364 New Functions In PHP 7.0

The following functions are now native functions in PHP 7.0. It is advised to change them before moving to this new version.

- get_resources()
- gc_mem_caches()
- preg_replace_callback_array()
- posix_setrlimit()
- random_bytes()
- random_int()
- intdiv()
- error_clear_last()

Short name	Php/Php70NewFunctions
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.365 New Functions In PHP 7.1

The following functions are now native functions in PHP 7.1. It is advised to change them before moving to this new version.

- curl_share_strerror()
- curl_multi_errno()
- curl_share_errno()
- mb_ord()
- mb_chr()
- mb_scrub()
- is_iterable()

Short name	Php/Php71NewFunctions
Rulesets	*CompatibilityPHP71*
Php Version	With PHP 7.1 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.366 New Functions In PHP 7.2

The following functions are now native functions in PHP 7.2. It is advised to change custom functions that are currently created, and using those names, before moving to this new version.

- mb_ord()
- mb_chr()
- mb_scrub()
- stream_isatty()
- proc_nice() (Windows only)

9.366.1 Suggestions

- Move custom functions with the same name to a new namespace
- Change the name of any custom functions with the same name
- Add a condition to the functions definition to avoid conflict

Short name	Php/Php72NewFunctions
Rulesets	*CompatibilityPHP72*
Php Version	With PHP 7.2 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.367 New Functions In PHP 7.3

New functions are added to new PHP version.

The following functions are now native functions in PHP 7.3. It is compulsory to rename any custom function that was created in older versions. One alternative is to move the function to a custom namespace, and update the `use` list at the beginning of the script.

- net_get_interfaces
- gmp_binomial
- gmp_lcm
- gmp_perfect_power
- gmp_kronecker
- openssl_pkey_derive
- is_countable
- ldap_exop_refresh

Note : At the moment of writing, all links to the manual are not working.

Short name	Php/Php73NewFunctions
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP73*
Php Version	With PHP 7.3 and older
Severity	Major
Time To Fix	Quick (30 mins)

9.368 New Functions In PHP 7.4

New functions are added to new PHP version.

The following functions are now native functions in PHP 7.3. It is compulsory to rename any custom function that was created in older versions. One alternative is to move the function to a custom namespace, and update the `use` list at the beginning of the script.

- mb_str_split
- password_algos

Note : At the moment of writing, all links to the manual are not working.

Short name	Php/Php74NewFunctions
Rulesets	*CompatibilityPHP74*
Php Version	With PHP 7.3 and older
Severity	Major
Time To Fix	Quick (30 mins)

9.369 New Functions In PHP 8.0

New functions are added to new PHP version.

The following functions are now native functions in PHP 7.3. It is compulsory to rename any custom function that was created in older versions. One alternative is to move the function to a custom namespace, and update the `use` list at the beginning of the script.

- str_contains
- fdiv
- preg_last_error_msg

Note : At the moment of writing, all links to the manual are not working.

Short name	Php/Php80NewFunctions
Rulesets	*CompatibilityPHP74*
Php Version	With PHP 8.0 and older
Severity	Major
Time To Fix	Quick (30 mins)

9.370 Next Month Trap

Avoid using +1 month with strtotime().

strtotime() calculates the next month by incrementing the month number. For day number that do not exist from one month to the next, strtotime() fixes them by setting them in the next-next month.

This happens to January, March, May, July, August and October. January is also vulnerable for 29 (not every year), 30 and 31.

Avoid using '+1 month', and rely on 'first day of next month' or 'last day of next month' to extract the next month's name.

```
<?php

// Base date is October 31 => 10/31
// +1 month adds +1 to 10 => 11/31
// Since November 31rst doesn't exists, it is corrected to 12/01.
echo date('F', strtotime('+1 month',mktime(0,0,0,$i,31,2017))).PHP_EOL;

// Base date is October 31 => 10/31
echo date('F', strtotime('first day of next month',mktime(0,0,0,$i,31,2017))).PHP_EOL;

?>
```

See also It is the 31st again.

9.370.1 Suggestions

- Review strtotime() usage for month additions
- Use datetime() and other classes, not PHP native functions
- Use a external library, like carbon, to handle dates

Short name	Structures/NextMonthTrap
Rulesets	*Analyze*, *Top10*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)
Examples	*Contao*, *Edusoho*

9.371 No Append On Source

Do not append new elements to an array in a foreach loop. Since PHP 7.0, the array is still used as a source, and will be augmented, and used again.

```
<?php

// Relying on the initial copy
$a = [1];
$initial = $a;
foreach($initial as $v) {
    $a[] = $v + 1;
}

// Keep new results aside
$a = [1];
$tmp = [];
foreach($a as $v) {
    $tmp[] = $v + 1;
}
$a = array_merge($a, $tmp);
unset($tmp);

// Example, courtesy of Frederic Bouchery
// This is an infinite loop
$a = [1];
foreach($a as $v) {
    $a[] = $v + 1;
}

?>
```

Thanks to Frederic Bouchery for the reminder.

See also foreach and What will this code return? #PHP.

9.371.1 Suggestions

- Use a copy of the source, to avoid modifying it during the loop
- Store the new values in a separate storage

Short name	Structures/NoAppendOnSource
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.372 No Boolean As Default

Default values should always be set up with a constant name.

Class constants and constants improve readability when calling the methods or comparing values and statuses.

```php
<?php

const CASE_INSENSITIVE = true;
const CASE_SENSITIVE = false;

function foo($case_insensitive = true) {
    // doSomething()
}

// Readable code
foo(CASE_INSENSITIVE);
foo(CASE_SENSITIVE);


// unreadable code  : is true case insensitive or case sensitive ?
foo(true);
foo(false);

?>
```

See also FlagArgument and Clean code: The curse of a boolean parameter.

9.372.1 Suggestions

- Use constants or class constants to give value to a boolean literal
- When constants have been defined, use them when calling the code
- Split the method into two methods, one for each case

Short name	Functions/NoBooleanAsDefault
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*OpenConf*

9.373 No Choice

A conditional structure is being used, but both alternatives are the same, leading to the illusion of choice.

Either the condition is useless, and may be removed, or the alternatives need to be distinguished.

```php
<?php

if ($condition == 2) {
    doSomething();
} else {
    doSomething();
```

(continues on next page)

(continued from previous page)

```
}

$condition == 2 ?     doSomething() :     doSomething();

?>
```

9.373.1 Suggestions

- Remove the conditional, and call the expression directly
- Replace one of the alternative with a distinct call
- Remove the whole conditional : it may end up being useless

Short name	Structures/NoChoice
Rulesets	*Analyze*, *Top10*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)
Examples	*NextCloud*, *Zencart*

9.374 No Class As Typehint

Avoid using classes as typehint : always use interfaces. This way, different classes, or versions of classes may be passed as argument. The typehint is not linked to an implementation, but to signatures.

A class is needed when the object is with properties : interfaces do not allow the specifications of properties.

```
<?php

class X {
    public $p = 1;

    function foo() {}
}

interface i {
    function foo();
}

// X is a class : any update in the code requires changing / subclassing X or the␣
↪rest of the code.
function bar(X $x) {
    $x->foo();
}

// I is an interface : X may implements this interface without refactoring and pass
// later, newer versions of X may get another name, but still implement I, and still␣
↪pass
function bar2(I $x) {
    $x->foo();
}
```

(continues on next page)

(continued from previous page)

```
function bar3(I $x) {
    echo $x->p;
}

?>
```

See also Type hinting for interfaces.

9.374.1 Suggestions

- Create an interface with the important methods, and use that interface
- Create an abstract class, when public properties are also needed

Short name	Functions/NoClassAsTypehint
Rulesets	*Typechecks*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Vanilla*, *phpMyAdmin*

9.375 No Class In Global

Avoid defining structures in Global namespace. Always prefer using a namespace. This will come handy later, either when publishing the code, or when importing a library, or even if PHP reclaims that name.

```
<?php

// Code prepared for later
namespace Foo {
    class Bar {}
}

// Code that may conflict with other names.
namespace {
    class Bar {}
}

?>
```

9.375.1 Suggestions

- Use a specific namespace for your classes

Short name	Php/NoClassInGlobal
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*Dolphin*

9.376 No Count With 0

Comparing count(), strlen() or mb_strlen() to 0 is a waste of resources. There are three distinct situations.

When comparing count() with 0, with ===, ==, !==, !=, it is more efficient to use empty(). empty() is a language construct that checks if a value is present, while count() actually load the number of element.

```
<?php

// Checking if an array is empty
if (count($array) == 0) {
    // doSomething();
}
// This may be replaced with
if (empty($array)) {
    // doSomething();
}

?>
```

When comparing count() strictly with 0 and >, it is more efficient to use `!(empty( ))`

```
<?php

// Checking if an array is empty
if (count($array) > 0) {
    // doSomething();
}
// This may be replaced with
if (!empty($array)) {
    // doSomething();
}

Of course comparing count() with negative values, or with >= is useless.

<?php

// Checking if an array is empty
if (count($array) < 0) {
    // This never happens
    // doSomething();
}

?>
```

Comparing count(), strlen() or mb_strlen() with other values than 0 cannot be replaced with a comparison with empty().

Note that this is a micro-optimisation : since PHP keeps track of the number of elements in arrays (or number of chars in strings), the total computing time of both operations is often lower than a ms. However, both functions tends to be heavily used, and may even be used inside loops.

See also count, strlen and mb_strlen.

9.376.1 Suggestions

- Use empty() on the data
- Compare the variable with a default value, such as an empty array

Short name	Performances/NotCountNull
Rulesets	*Performances*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Contao*, *WordPress*

9.377 No Direct Call To Magic Method

PHP features magic methods, which are methods related to operators.

Magic methods, such as __get(), related to =, or __clone(), related to `clone`, are supposed to be used in an object environment, and not with direct call.

It is recommended to use the magic method with its intended usage, and not to call it directly. For example, typecast to `string` instead of calling the __toString() method.

```
<?php
// Write
  print $x->a;
// instead of
  print $x->__get('a');

class Foo {
    private $b = secret;

    public function __toString() {
        return strtoupper($this->b);
    }
}

$bar = new Foo();
echo (string) $bar;

?>
```

Accessing those methods in a static way is also discouraged.

See also Magic Methods and Magical PHP: '__call <https://www.garfieldtech.com/blog/magical-php-call>'_.

Short name	Classes/DirectCallToMagicMethod
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)

9.378 No Direct Usage

The results of the following functions shouldn't be used directly, but checked first.

For example, glob() returns an array, unless some error happens, in which case it returns a boolean (false). In such case, however rare it is, plugging glob() directly in a foreach() loops will yield errors.

```
<?php
    // Used without check :
    foreach(glob('.') as $file) { /* do Something */ }.

    // Used without check :
    $files = glob('.');
    if (!is_array($files)) {
        foreach($files as $file) { /* do Something */ }.
    }
?>
```

9.378.1 Suggestions

- Check the return of the function before using it, in particular for false, or array().

Short name	Structures/NoDirectUsage
Rulesets	*Analyze*
Severity	Major
Time To Fix	Slow (1 hour)
Examples	*Edusoho*, *XOOPS*

9.379 No ENT_IGNORE

Certain characters have special significance in HTML, and should be represented by HTML entities if they are to preserve their meanings.

ENT_IGNORE is a configuration option for htmlspecialchars(), that ignore any needed character replacement. This mean the raw input will now be processed by PHP, or a target browser.

It is recommended to use the other configuration options : `ENT_COMPAT`, `ENT_QUOTES`, `ENT_NOQUOTES`, `ENT_SUBSTITUTE`, `ENT_DISALLOWED`, `ENT_HTML401`, `ENT_XML1`, `ENT_XHTML` or `ENT_HTML5`.

```
<?php

// This produces a valid HTML tag
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_IGNORE);
echo $new; // &lt;a href=&#039;test&#039;&gt;Test&lt;/a&gt;

// This produces a valid string, without any HTML special value
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
echo $new; // &lt;a href=&#039;test&#039;&gt;Test&lt;/a&gt;

?>
```

See also htmlspecialchars and Deletion of Code Points.

9.379.1 Suggestions

- Use of the the other options

Short name	Security/NoEntIgnore
Rulesets	*Security*
Severity	Minor
Time To Fix	Quick (30 mins)

9.380 No Empty Regex

PHP regex don't accept empty regex, nor regex with alphanumeric delimiter.

Most of those errors happen at execution time, when the regex is build dynamically, but still may end empty. At compile time, such error are made when the code is not tested before commit.

```
<?php

// No empty regex
preg_match('', $string, $r);

// Delimiter must be non-alphanumerical
preg_replace('1abc1', $string, $r);

// Delimiter must be non-alphanumerical
preg_replace('1'.$regex.'1', $string, $r);

?>
```

See also PCRE and Delimiters.

9.380.1 Suggestions

- Fix the regex by adding regex delimiters

Short name	Structures/NoEmptyRegex
Rulesets	*Analyze*, *CI-checks*
Severity	Critical
Time To Fix	Quick (30 mins)
Examples	*Tikiwiki*

9.381 No Hardcoded Hash

Hash should never be hardcoded.

Hashes may be MD5, SHA1, SHA512, Bcrypt or any other. Such values must be easily changed, for security reasons, and the source code is not the safest place to hide it.

```
<?php

    // Those strings may be sha512 hashes.
    // it is recomemdned to check if they are static or should be put into␣
↪configuration
    $init512 = array( // initial values for SHA512
```

(continues on next page)

(continued from previous page)

```
        '6a09e667f3bcc908', 'bb67ae8584caa73b', '3c6ef372fe94f82b', 'a54ff53a5f1d36f1
↪',
    );

    // strings which are obvious conversion are ignored
    $decimal = intval('87878877', 12);
?>
```

See also Salted Password Hashing - Doing it Right and Hash-Buster.

9.381.1 Suggestions

- Put any hardcoded hash in a configuration file, a database or a environment variable. An external source.

Short name	Structures/NoHardcodedHash
Rulesets	*Analyze*, *Security*
Severity	Critical
Time To Fix	Slow (1 hour)
Examples	*shopware*, *SugarCrm*

9.382 No Hardcoded Ip

Do not leave hard coded IP in your code.

It is recommended to move such configuration in external files or databases, for each update. This may also come handy when testing.

```
<?php

// This IPv4 is hardcoded.
$ip = '183.207.224.50';
// This IPv6 is hardcoded.
$ip = '2001:0db8:85a3:0000:0000:8a2e:0370:7334';

// This looks like an IP
$thisIsNotAnIP = '213.187.99.50';
$thisIsNotAnIP = '2133:1387:9393:5330';

?>
```

`127.0.0.1`, `\:\:1` and `\:\:0` are omitted, and not considered as a violation.

See also Use of Hardcoded IPv4 Addresses and Never hard code sensitive information.

9.382.1 Suggestions

- Move the hardcoded IP to an external source : environment variable, configuration file, database.
- Remove the hardcoded IP and ask for it at execution.
- Use a literal value for default messages in form.

Short name	Structures/NoHardcodedIp
Rulesets	*Analyze*, *Security*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*OpenEMR*, *NextCloud*

9.383 No Hardcoded Path

It is not recommended to use hardcoded literals when designating files. Full paths are usually tied to one file system organization. As soon as the organisation changes or must be adapted to any external constraint, the path is not valid anymore.

Either use __FILE__ and __DIR__ to make the path relative to the current file; use a `DOC_ROOT` as a configuration constant that will allow the moving of the script to another folder; finally functions like sys_get_temp_dir() produce a viable temporary folder.

Relative paths are relative to the current execution directory, and not the current file. This means they may differ depending on the location of the start of the application, and are sensitive to chdir() and chroot() usage.

```php
<?php

    // This depends on the current executed script
    file_get_contents('token.txt');

    // Exotic protocols are ignored
    file_get_contents('jackalope://file.txt');

    // Some protocols are ignored : http, https, ftp, ssh2, php (with memory)
    file_get_contents('http://www.php.net/');
    file_get_contents('php://memory/');

    // glob() with special chars * and ? are not reported
    glob('./*/foo/bar?.txt');
    // glob() without special chars * and ? are reported
    glob('/foo/bar/');

?>
```

9.383.1 Suggestions

- Add __DIR__ before the path to make it relative to the current file
- Add a configured prefix before the path to point to any file in the system
- Use sys_get_temp_dir() for temporary data
- Use `include_path` argument function, such as fie_get_contents(), to have the file located in configurable directories.

Short name	Structures/NoHardcodedPath
Rulesets	*Analyze*
Severity	Major
Time To Fix	Slow (1 hour)
ClearPHP	no-hardcoded-path
Examples	*Tine20*, *Thelia*

9.384 No Hardcoded Port

When connecting to a remove server, port is an important information. It is recommended to make this configurable (with constant or configuration), to as to be able to change this value without changing the code.

```
<?php

    // Both configurable IP and hostname
    $connection = ssh2_connect($_ENV['SSH_HOST'], $_ENV['SSH_PORT'], $methods,
↪$callbacks);

    // Both hardcoded IP and hostname
    $connection = ssh2_connect('shell.example.com', 22, $methods, $callbacks);

    if (!$connection) die('Connection failed');
?>
```

9.384.1 Suggestions

- Move the port to a configuration file, an environment variable

Short name	Structures/NoHardcodedPort
Rulesets	*Analyze*, *Security*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*WordPress*

9.385 No List With String

list() can't be used anymore to access particular offset in a string. This should be done with substr() or $string[$offset] syntax.

```
<?php

$x = 'abc';
list($a, $b, $c) = $x;

//list($a, $b, $c) = 'abc'; Never works

print $c;
// PHP 5.6- displays 'c'
// PHP 7.0+ displays nothing
```

(continues on next page)

(continued from previous page)

```
?>
```

See also PHP 7.0 Backward incompatible changes : list() can no longer unpack string variables .

9.385.1 Suggestions

- Use str_split() to break a string into bytes
- Use substr() or $string[$offset] syntax to access specific bytes in the string

Short name	Php/NoListWithString
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Instant (5 mins)

9.386 No Literal For Reference

Method arguments and return values may be by reference. Then, they need to be a valid variable.

Objects are always passed by reference, so there is no need to explicitly declare it.

Expressions, including ternary operator, produce value, and can't be used by reference directly. This is also the case for expression that include one or more reference.

```
<?php

// variables, properties, static properties, array items are all possible
$a = 1;
foo($a);

//This is not possible, as a literal can't be a reference
foo(1);

function foo(&$int) { return $int; }


// This is not a valid reference
function &bar() { return 2; }
function &bar2() { return 2 + $r; }

?>
```

Wrongly passing a value as a reference leads to a PHP Notice.

See also References.

9.386.1 Suggestions

- Remove the reference in the method signature (argument or return value)
- Make the argument an object, by using a typehint (non-scalar)

- Put the value into a variable prior to call (or return) the method

Short name	Functions/NoLiteralForReference
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.387 No Magic With Array

Magic method `__set()` doesn't work for array syntax.

When overloading properties, they can only be used for scalar values, excluding arrays. Under the hood, PHP uses `__get()` to reach for the name of the property, and doesn't recognize the following index as an array. It yields an error : Indirect modification of overloaded property.

```php
<?php

class c {
    private $a;
    private $o = array();

    function __get($name) {
        return $this->o[$name];
    }

    function foo() {
        // property b doesn't exists
        $this->b['a'] = 3;

        print_r($this);
    }

    // This method has no impact on the issue
    function __set($name, $value) {
        $this->o[$name] = $value;
    }
}

$c = new c();
$c->foo();

?>
```

It is possible to use the array syntax with a magic property : by making the `__get` returns an array, the syntax will actually extract the expected item in the array.

This is not reported by linting.

In this analysis, only properties that are found to be magic are reported. For example, using the b property outside the class scope is not reported, as it would yield too many false-positives.

See also Overload.

9.387.1 Suggestions

- Use a distinct method to append a new value to that property
- Assign the whole array, and not just one of its elements

Short name	Classes/NoMagicWithArray
Rulesets	*Analyze*, *LintButWontExec*, *CI-checks*
Severity	Major
Time To Fix	Slow (1 hour)

9.388 No More Curly Arrays

Only use square brackets to access array elements. The usage of curly brackets for array access is deprecated since PHP 7.4.

```
<?php

$array = [1,2,3];

// always valid
echo $array[1];

// deprecated in PHP 7.4
echo $array{1};

?>
```

See also Deprecate curly brace syntax and Deprecate curly brace syntax for accessing array elements and string offsets.

9.388.1 Suggestions

- Always use square brackets to access particular index in an array

Short name	Php/NoMoreCurlyArrays
Rulesets	*CompatibilityPHP74*
Php Version	8.0-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

9.389 No Need For Else

Else is not needed when the Then ends with a break. A break may be the following keywords : break, continue, return, goto. Any of these send the execution somewhere in the code. The else block is then executed as the main sequence, only if the condition fails.

```
<?php

function foo() {
```

(continues on next page)

(continued from previous page)

```
    // Else may be in the main sequence.
    if ($a1) {
        return $a1;
    } else {
        $a++;
    }

    // Same as above, but negate the condition : if (!$a2) { return $a2; }
    if ($a2) {
        $a++;
    } else {
        return $a2;
    }

    // This is OK
    if ($a3) {
        return;
    }

    // This has no break
    if ($a4) {
        $a++;
    } else {
        $b++;
    }

    // This has no else
    if ($a5) {
        $a++;
    }
}
?>
```

See also Object Calisthenics, rule # 2.

9.389.1 Suggestions

- Remove else block, but keep the code

Short name	Structures/NoNeedForElse
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Thelia*, *ThinkPHP*

9.390 No Need For Triple Equal

There is no need for the identity comparison when the methods returns the proper type.

```
<?php

// foo() returns a string.
```

(continues on next page)

(continued from previous page)

```
if ('a' === foo()) {
    // doSomething()
}


function foo() : string {
    return 'a';
}

?>
```

9.390.1 Suggestions

-

Short name	Structures/NoNeedForTriple
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.391 No Need For get_class()

There is no need to call get_class() to build a static call. The argument of get_class() may be used directly.

```
<?php

//
$a->b::$c

// This is too much code
get_class($a->b)::$c

?>
```

See also Scope Resolution Operator (::).

9.391.1 Suggestions

- Use get_called_class(), which may carry different class names
- Use self, static or parent keywords, if you are already in the current class
- Use the argument of get_class() directly

Short name	Structures/NoNeedGetClass
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.392 No Net For Xml Load

Simplexml and ext/DOM load all external entities from the web, by default. This is dangerous, in particular when loading unknown XML code.

Look at this XML code below : it is valid. It defines an entity `xxe`, that is filled with a file, read on the system and base64 encoded.:

```
&lt;!DOCTYPE replace [&lt;!ENTITY xxe SYSTEM "php://filter/convert.base64-encode/
↪resource=index.php"&gt; ]&gt;
<replace>&xxe;</replace>
```

This file could be processed with the following code : note, you can replace 'index.php' in the above entity by any valid filepath.

```
<?php
    $dom = new DOMDocument();
    $dom->loadXML($xml, LIBXML_NOENT | LIBXML_DTDLOAD);
    $info = simplexml_import_dom($dom);

    print base64_decode($info[0]);
?>
```

Here, PHP tries to load the XML file, finds the entity, then solves the entity by encoding a file called `index.php`. The source code of the file is not used as data in the XML file.

At that point, the example illustrates how a XXE works : by using the XML engine to load external resources, and preprocessing the XML code. in fact, there is only one change to make this XML code arbitrarily injected ::

```
&lt;!DOCTYPE replace [&lt;!ENTITY writer SYSTEM https://www.example.com/entities.dtd&
↪gt; ]&gt;
<replace>&xxe;</replace>
```

With the above example, the XML code is static (as, it never changes), but the 'xxe' definitions are loaded from a remove website, and are completely under the attacker control.

See also XML External Entity, XML External Entity (XXE) Processing and Detecting and exploiting XXE in SAML Interfaces.

9.392.1 Suggestions

- Strip out any entity when using external XML
- Forbid any network to the XML engine, by configuring the XML engine without network access

Short name	Security/NoNetForXmlLoad
Rulesets	*Security*
Severity	Major
Time To Fix	Slow (1 hour)

9.393 No Parenthesis For Language Construct

Some PHP language constructs, such are `include`, `print`, `echo` don't need parenthesis. They accept parenthesis, but it is may lead to strange situations.

```
<?php

// This is an attempt to load 'foo.inc', or kill the script
include('foo.inc') or die();
// in fact, this is read by PHP as : include 1
// include  'foo.inc' or die();

?>
```

It it better to avoid using parenthesis with `echo`, `print`, `return`, `throw`, `yield`, `yield from`, `include`, `require`, `include_once`, `require_once`.

See also include.

9.393.1 Suggestions

- Remove parenthesis

Short name	Structures/NoParenthesisForLanguageConstruct
Rulesets	*Analyze*, *Suggestions*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
ClearPHP	no-parenthesis-for-language-construct
Examples	*Phpdocumentor*, *phpMyAdmin*

9.394 No Plus One

Incrementing a variable should be done with the ++ or – operators. Any other way, may be avoided.

```
<?php

// Best way to increment
++$x; --$y;

// Second best way to increment, if the current value is needed :
echo $x++, $y--;

// Good but slow
$x += 1;
$x -= -1;

$y += -1;
$y -= 1;

// even slower
$x = $x + 1;
$y = $y - 1;

?>
```

Short name	Structures/PlusEgalOne
Rulesets	*Coding Conventions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.395 No Public Access

The properties below are declared with public access, but are never used publicly. They can be made protected or private.

```
<?php

class foo {
    public $bar = 1;            // Public, and used in public space
    public $neverInPublic = 3;  // Public, but never used in outside the class

    function bar() {
        $neverInPublic++;
    }
}

$x = new foo();
$x->bar = 3;
$x->bar();

?>
```

Short name	Classes/NoPublicAccess
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.396 No Real Comparison

Avoid comparing decimal numbers with ==, ===, !==, !=. Real numbers have an error margin which is random, and makes it very difficult to match even if the compared value is a literal.

PHP uses an internal representation in base 2 : any number difficult to represent with this base (like 0.1 or 0.7) will have a margin of error.

```
<?php

$a = 1/7;
$b = 2.0;

// 7 * $a is a real, not an integer
var_dump( 7 * $a === 1);

// rounding error leads to wrong comparison
var_dump( (0.1 + 0.7) * 10 == 8);
// although
```

(continues on next page)

(continued from previous page)

```
var_dump( (0.1 + 0.7) * 10);
// displays 8

// precision formula to use with reals. Adapt 0.0001 to your precision needs
var_dump( abs(((0.1 + 0.7) * 10) - 8) < 0.0001);

?>
```

Use precision formulas with abs() to approximate values with a given precision, or avoid reals altogether.

See also Floating point numbers.

9.396.1 Suggestions

- Cast the values to integer before comparing
- Compute the difference, and keep it below a threshold
- Use the gmp or the bc extension to handle high precision numbers
- Change the 'precision' directive of PHP : ini_set('precision', 30) to make number larger
- Multiply by a power of ten, before casting to integer for the comparison
- Use floor(), ceil() or round() to compare the numbers, with a specific precision

Short name	Type/NoRealComparison
Rulesets	*Analyze*, *Top10*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	no-real-comparison
Examples	*Magento*, *SPIP*

9.397 No Reference For Static Property

Static properties used to behave independently when set to a reference value. This was fixed in PHP 7.3.

According to the PHP 7.3 changelog : `In PHP, `static <https://www.php.net/manual/en/language.oop5.static.php>`_ properties are shared between inheriting classes, unless the `static <https://www.php.net/manual/en/language.oop5.static.php>`_ property is explicitly overridden in a child class. However, due to an implementation artifact it was possible to separate the `static <https://www.php.net/manual/en/language.oop5.static.php>`_ properties by assigning a reference. This loophole has been fixed..`

```
<?php

        class Test {
            public static $x = 0;
        }
        class Test2 extends Test { }

        Test2::$x = &$x;
        $x = 1;
```

(continues on next page)

(continued from previous page)

```
        var_dump(Test::$x, Test2::$x);
        // Previously: int(0), int(1)
        // Now: int(1), int(1)

?>
```

See also PHP 7.3 UPGRADE NOTES.

Short name	Php/NoReferenceForStaticProperty
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.3 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.398 No Reference For Ternary

The ternary operator and the null coalescing operator are both expressions that only return values, and not a variable.

This means that any provided reference will be turned into its value. While this is usually invisible, it will raise a warning when a reference is expected. This is the case with methods returning a reference.

A PHP notice is generated when using a ternary operator or the null coalesce operator : `Only variable references should be returned by reference`. The notice is also emitted when returning objects.

This applies to methods, functions and closures.

```
<?php

// This works
function &foo($a, $b) {
    if ($a === 1) {
        return $b;
    } else {
        return $a;
    }
}

// This raises a warning, as the operator returns a value
function &foo($a, $b) { return $a === 1 ? $b : $a; }

?>
```

See also Null Coalescing Operator, Ternary Operator.

9.398.1 Suggestions

- Drop the reference at assignation time

- Drop the reference in the argument definition
- Drop the reference in the function return definition

Short name	Php/NoReferenceForTernary
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Slow (1 hour)
Examples	*phpadsnew*

9.399 No Reference On Left Side

Do not use references as the right element in an assignation.

```
<?php

$b = 2;
$c = 3;

$a = &$b + $c;
// $a === 2 === $b;

$a = $b + $c;
// $a === 5

?>
```

This is the case for most situations : addition, multiplication, bitshift, logical, power, concatenation. Note that PHP won't compile the code if the operator is a short operator (+=, .=, etc.), nor if the & is on the right side of the operator.

Short name	Structures/NoReferenceOnLeft
Rulesets	*Analyze*, *CI-checks*
Severity	Critical
Time To Fix	Quick (30 mins)

9.400 No Return For Generator

Return is not allowed in generator. In PHP versions older than 5.6 and older, they yield a fatal Error.

```
<?php

function generatorWithReturn() {
    yield 1;
    return 2;
}

?>
```

See also Generators overview.

Short name	Php/NoReturnForGenerator
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and more recent
Severity	Critical
Time To Fix	Quick (30 mins)

9.401 No Return Or Throw In Finally

Avoid using return and throw in a finally block. Both command will interrupt the processing of the try catch block, and any exception that was emitted will not be processed. This leads to unprocessed exceptions, leaving the application in an unstable state.

Note that PHP prevents the usage of goto, break and continue within the finally block at linting phase. This is categorized as a Security problem.

```php
<?php
function foo() {
        try {
            // Exception is thrown here
            throw new \Exception();
        } catch (Exception $e) {
            // This is executed AFTER finally
            return 'Exception';
        } finally {
            // This is executed BEFORE catch
            return 'Finally';
        }
    }
}

// Displays 'Finally'. No exception
echo foo();

function bar() {
        try {
            // Exception is thrown here
            throw new \Exception();
        } catch (Exception $e) {
            // Process the exception.
            return 'Exception';
        } finally {
            // clean the current situation
            // Keep running the current function
        }
        return 'Finally';
    }
}

// Displays 'Exception', with processed Exception
echo bar();

?>
```

See also Return Inside Finally Block.

9.401.1 Suggestions

- Move the return right after the try/catch/finally call

Short name	Structures/NoReturnInFinally
Rulesets	*Security*
Severity	Major
Time To Fix	Quick (30 mins)

9.402 No Return Used

The return value of the following functions are never used. The return argument may be dropped from the code, as it is dead code.

This analysis supports functions and static methods, when a definition may be found. It doesn't support method calls.

```
<?php

function foo($a = 1;) { return 1; }
foo();
foo();
foo();
foo();
foo();
foo();

// This function doesn't return anything.
function foo2() { }

// The following function are used in an expression, thus the return is important
function foo3() {  return 1;}
function foo4() {  return 1;}
function foo5() {  return 1;}

foo3() + 1;
$a = foo4();
foo(foo5());

?>
```

9.402.1 Suggestions

- Remove the return statement in the function
- Actually use the value returned by the method, for test or combination with other values

Short name	Functions/NoReturnUsed
Rulesets	*Analyze*, *Suggestions*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*SPIP*, *LiveZilla*

9.403 No Self Referencing Constant

It is not possible to use a constant to define itself in a class. It yields a fatal error at runtime.

The PHP error reads : Cannot declare `self <https://www.php.net/manual/en/language.oop5.paamayim-nekudotayim.php>`_-referencing constant '`self <https://www.php.net/manual/en/language.oop5.paamayim-nekudotayim.php>`_\:\:C2'. Unlike PHP which is self-referencing, self referencing variables can't have a value : just don't use that.

```
<?php
    class a {
        const C1 = 1;         // fully defined constant
        const C2 = self::C2;  // self referencing constant
        const C3 = a::C3 + 2; // self referencing constant
    }
?>
```

The code may access an already declared constant with self or with its class name.

```
<?php
    class a {
        const C1 = 1;
        const C2 = a::C1;
    }
?>
```

This error is not detected by linting. It is only detected at instantiation time : if the class is not used, it won't appear.

9.403.1 Suggestions

- Give a literal value to this constant
- Give a constant value to this constant : other class constants or constant are allowed here.

Short name	Classes/NoSelfReferencingConstant
Rulesets	*Analyze*, *LintButWontExec*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.404 No Spread For Hash

The spread operator ... only works on integer-indexed arrays.

```
<?php

// This is valid, as ``-33`` is cast to integer by PHP automagically
var_dump(...[1,-33 => 2, 3]);

// This is not valid
var_dump(...[1,C => 2, 3]);

?>
```

See also Variable-length argument lists.

9.404.1 Suggestions

- Add a call to array_values() instead of the hash

Short name	Arrays/NoSpreadForHash
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.405 No String With Append

PHP 7 doesn't allow the usage of [] with strings. [] is an array-only operator.

```
<?php

$string = 'abc';

// Not possible in PHP 7
$string[] = 'd';

?>
```

This was possible in PHP 5, but is now forbidden in PHP 7.

9.405.1 Suggestions

- Use the concatenation operator `.` to append strings.
- Use the concatenation short assignement `.=` to append strings.

Short name	Php/NoStringWithAppend
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and more recent
Severity	Major
Time To Fix	Quick (30 mins)

9.406 No Substr Minus One

Negative index were introduced in PHP 7.1. This syntax is not compatible with PHP 7.0 and older.

```
<?php
$string = 'abc';

echo $string[-1]; // c

echo $string[1]; // a

?>
```

See also Generalize support of negative string offsets.

9.406.1 Suggestions

- Use the -1 index in a string, instead of a call to substr()

Short name	Php/NoSubstrMinusOne
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.1 and more recent
Severity	Major
Time To Fix	Quick (30 mins)

9.407 No Weak SSL Crypto

When enabling PHP's stream SSL, it is important to use a safe protocol.

All the SSL protocols (1.0, 2.0, 3.0), and TLS (1.0 are unsafe. The best is to use the most recent TLS, version 1.2.

stream_socket_enable_crypto() and curl_setopt() are checked.

```
<?php

// This socket will use SSL v2, which
$socket = 'sslv2://www.example.com';
$fp = fsockopen($socket, 80, $errno, $errstr, 30);

?>
```

Using the TLS transport protocol of PHP will choose the version by itself.

See also Insecure Transportation Security Protocol Supported (TLS 1.0), The 2018 Guide to Building Secure PHP Software and Internet Domain: TCP, UDP, SSL, and TLS.

9.407.1 Suggestions

- Use TLS transport, with version 1.2

Short name	Security/NoWeakSSLCrypto
Rulesets	*Security*
Severity	Minor
Time To Fix	Quick (30 mins)

9.408 No array_merge() In Loops

array_merge() is memory intensive : every call will duplicate the arguments in memory, before merging them.

To handle arrays that may be quite big, it is recommended to avoid using array_merge() in a loop. Instead, one should use array_merge() with as many arguments as possible, making the merge a on time call.

```
<?php

// A large multidimensional array
$source = ['a' => ['a', 'b', /*...*/],
           'b' => ['b', 'c', 'd', /*...*/],
           /*...*/
           ];

// Faster way
$b = array();
foreach($source as $key => $values) {
    //Collect in an array
    $b[] = $values;
}

// One call to array_merge
$b = call_user_func_array('array_merge', $b);
// or with variadic
$b = call_user_func('array_merge', ..$b);

// Fastest way (with above example, without checking nor data pulling)
$b = call_user_func_array('array_merge', array_values($source))
// or
$b = call_user_func('array_merge', ...array_values($source))

// Slow way to merge it all
$b = array();
foreach($source as $key => $values) {
    $b = array_merge($b, $values);
}

?>
```

Note that array_merge_recursive() and file_put_contents() are affected and reported the same way.

9.408.1 Suggestions

- Store all intermediate arrays in a temporary variable, and use array_merge() once, with ellipsis or call_user_func_array().

Short name	Performances/ArrayMergeInLoops
Rulesets	*Analyze*, *Performances*, *Top10*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	no-array_merge-in-loop
Examples	*Tine20*

9.409 No get_class() With Null

It is not possible to pass explicitly null to get_class() to get the current's class name. Since PHP 7.2, one must call get_class() without arguments to achieve that result.

```
<?php

class A {
  public function f() {
    // Gets the classname
    $classname = get_class();

    // Gets the classname and a warning
    $classname = get_class(null);
  }
}

$a = new A();
$a->f('get_class');

?>
```

Short name	Structures/NoGetClassNull
Rule-sets	*Analyze*, *CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP72*
Severity	Major
Time To Fix	Instant (5 mins)

9.410 No isset() With empty()

empty() actually does the job of isset() too.

From the manual : `No warning is generated if the variable does not exist. That means `empty() <https://www.php.net/empty>`_ is essentially the concise equivalent to !`isset(<https://www.www.php.net/isset>`_$var) || $var == false`. The main difference is that isset() only works with variables, while empty() works with other structures, such as constants.

```
<?php


// Enough validation
if (!empty($a)) {
    doSomething();
}

// Too many tests
if (isset($a) && !empty($a)) {
    doSomething();
}

?>
```

See also Isset <http://www.php.net/'isset>'_ and empty.

9.410.1 Suggestions

- Only use isset(), just drop the empty()
- Only use empty(), just drop the empty()
- Use a null value, so the variable is always set

Short name	Structures/NoIssetWithEmpty
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*XOOPS*

9.411 No mb_substr In Loop

Do not use loops on mb_substr().

mb_substr() always starts at the beginning of the string ot search for the nth char, and recalculate everything. This means that the first iterations are as fast as substr() (for comparison), while the longer the string, the slower mb_substr().

The recommendation is to use preg_split() with the *u* option, to split the string into an array. This save multiple recalculations.

```
<?php

// Split the string by characters
$array = preg_split('//u', $string, -1, PREG_SPLIT_NO_EMPTY);
foreach($array as $c) {
    doSomething($c);
}

// Slow version
$nb = mb_strlen($mb);
for($i = 0; $i < $nb; ++$i) {
    // Fetch a character
    $c = mb_substr($string, $i, 1);
    doSomething($c);
}

?>
```

See also Optimization: How I made my PHP code run 100 times faster and How to iterate UTF-8 string in PHP?.

9.411.1 Suggestions

- Use preg_split() and loop on its results.

Short name	Performances/MbStringInLoop
Rulesets	*Performances*
Severity	Minor
Time To Fix	Quick (30 mins)

9.412 Non Ascii Variables

PHP allows certain characters in variable names. The variable name must only include letters, figures, underscores and ASCII characters from 128 to 255.

In practice, letters outside the scope of `a-zA-Z0-9` are rare, and require more care when editing the code or passing it from OS to OS.

```
<?php

class  {
    // An actual working class in PHP.
    public function __construct() {
        echo __CLASS__;
    }
}

$ = new ();

?>
```

See also Variables.

9.412.1 Suggestions

- Make sure those special chars have actual meaning.

Short name	Variables/VariableNonascii
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*Magento*

9.413 Non Nullable Getters

A getter needs to be nullable when a property is injected.

In particular, if the injection happens with a separate method, there is a time where the object is not consistent, and the property holds a default non-object value.

```
<?php

class Consistent {
    private $db = null;

    function __construct(Db $db) {
        $this->db = $db;
        // Object is immediately consistent
    }

    // Db might be null
    function getDb() {
        return $this->db;
```

(continues on next page)

(continued from previous page)

```
    }
}

class Inconsistent {
    private $db = null;

    function __construct() {
        // No initialisation
    }

    // This might be called on time, or not
    // This typehint cannot be nullable, nor use null as default
    function setDb(DB $db) {
        return $this->db;
    }

    // Db might be null
    function getDb() {
        return $this->db;
    }
}
?>
```

9.413.1 Suggestions

- Remove the nullable option and the tests on `null`.

Short name	Classes/NonNullableSetters
Rulesets	*Analyze*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.414 Non Static Methods Called In A Static

Static methods have to be declared as such (using the static keyword). Then, one may call them without instantiating the object.

PHP 7.0, and more recent versions, yield a deprecated error : `` Non-`static <https://www.php.net/manual/en/language.oop5.static.php>`_ method A\:\:B() should not be called statically ``.

PHP 5 and older doesn't check that a method is static or not : at any point, the code may call one method statically.

```
<?php
    class x {
        static public function sm( ) { echo __METHOD__.\n; }
        public public sm( ) { echo __METHOD__.\n; }
    }

    x::sm( ); // echo x::sm

    // Dynamic call
```

(continues on next page)

(continued from previous page)

```
    ['x', 'sm']();
    [\x::class, 'sm']();

    $s = 'x::sm';
    $s();

?>
```

It is a bad idea to call non-static method statically. Such method may make use of special variable $this, which will be undefined. PHP will not check those calls at compile time, nor at running time.

It is recommended to update this situation : make the method actually static, or use it only in object context.

Note that this analysis reports all static method call made on a non-static method, even within the same class or class hierarchy. PHP silently accepts static call to any in-family method.

```
<?php
    class x {
        public function foo( ) { self::bar() }
        public function bar( ) { echo __METHOD__.\n; }
    }
?>
```

See also Static Keyword <https://www.php.net/manual/en/language.oop5.‘static.php>‘_.

9.414.1 Suggestions

- Call the method the correct way
- Define the method as static

Short name	Classes/NonStaticMethodsCalledStatic
Rulesets	*Analyze*, *CompatibilityPHP56*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Dolphin*, *Magento*

9.415 Non-constant Index In Array

Undefined constants revert as strings in Arrays. They are also called `barewords`.

In `$array[index]`, PHP cannot find index as a constant, but, as a default behavior, turns it into the string `index`.

This default behavior raise concerns when a corresponding constant is defined, either using define() or the const keyword (outside a class). The definition of the index constant will modify the behavior of the index, as it will now use the constant definition, and not the ‘index’ string.

```
<?php

// assign 1 to the element index in $array
// index will fallback to string
```

(continues on next page)

(continued from previous page)

```
$array[index] = 1;
//PHP Notice:  Use of undefined constant index - assumed 'index'

echo $array[index];      // display 1 and the above error
echo "$array[index]";    // display 1
echo "$array['index']";  // Syntax error


define('index', 2);

 // now 1 to the element 2 in $array
 $array[index] = 1;

?>
```

It is recommended to make index a real string (with ‘ or “), or to define the corresponding constant to avoid any future surprise.

Note that PHP 7.2 removes the support for this feature.

See also PHP RFC: Deprecate and Remove Bareword (Unquoted) Strings and Syntax.

9.415.1 Suggestions

- Declare the constant to give it an actual value
- Turn the constant name into a string

Short name	Arrays/NonConstantArray
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Dolibarr*, *Zencart*

9.416 Non-lowercase Keywords

The usual convention is to write PHP keywords (like `as`, `foreach`, `switch`, `case`, `break`, etc.) all in lowercase.

```
<?php

// usual PHP convention
foreach($array as $element) {
    echo $element;
}

// unusual PHP conventions
Foreach($array AS $element) {
    eCHo $element;
}

?>
```

PHP understands them in lowercase, UPPERCASE or WilD Case, so there is nothing compulsory here. Although, it will look strange to many.

Some keywords are missing from this analysis : `extends`, `implements`, `as`. This is due to the internal engine, which doesn't keep track of them in its AST representation.

9.416.1 Suggestions

- Use lowercase only PHP keywords, except for constants such as __CLASS__.

Short name	Php/UpperCaseKeyword
Rulesets	*Coding Conventions*
Severity	Minor
Time To Fix	Instant (5 mins)

9.417 Not A Scalar Type

`int` is the actual PHP scalar type, not `integer`.

PHP 7 introduced several scalar types, in particular `int`, `bool` and `float`. Those three types are easily mistaken with `integer`, `boolean`, `real` and `double`.

Unless those classes actually exists, PHP emits some strange error messages.

```php
<?php

// This expects a scalar of type 'integer'
function foo(int $i) {}

// This expects a object of class 'integer'
function abr(integer $i) {}

?>
```

Thanks to `Benoit Viguier` for the original idea for this analysis.

See also Type declarations.

9.417.1 Suggestions

- Do not use `int` as a class name, an interface name or a trait name.

Short name	Php/NotScalarType
Rulesets	*Typechecks*
Severity	Major
Time To Fix	Instant (5 mins)

9.418 Not Equal Is Not !==

Not and Equal operators, used separately, don't amount to the different operator `!==`.

`!$a == $b` first turns `$a``into the opposite boolean, then compares this boolean value to ``$b`. On the other hand, `$a !== $b` compares the two variables for type and value, and returns a boolean.

```
<?php

if ($string != 'abc') {
    // doSomething()
}

// Here, string will be an boolean, leading
if (!$string == 'abc') {
    // doSomething()
}

// operator priority may be confusing
if (!$object instanceof OneClass) {
    // doSomething()
}
?>
```

Note that the `instanceof` operator may be use with this syntax, due to operator precedence.

See also Operator Precedence.

9.418.1 Suggestions

- Use the != or !==
- Use parenthesis

Short name	Structures/NotEqual
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.419 Not Not

Double not makes a boolean, not a `true`.

This is a wrong casting to boolean. PHP supports `(boolean)` to do the same, faster and cleaner.

```
<?php
    // Explicit code
    $b = (boolean) $x;
    $b = (bool) $x;

    // Wrong type casting
    $b = !!$x;

?>
```

See also Logical Operators and Type Juggling.

9.419.1 Suggestions

- Use `(bool)` casting operator for that

- Don't typecast, and let PHP handle it. This works in situations where the boolean is immediately used.

Short name	Structures/NotNot
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
ClearPHP	no-implied-cast
Examples	*Cleverstyle*, *Tine20*

9.420 Null On New

Until PHP 7, some classes instantiation could yield null, instead of throwing an exception.

After issuing a 'new' with those classes, it was important to check if the returned object were null or not. No exception were thrown.

```php
<?php

// Example extracted from the wiki below
$mf = new MessageFormatter('en_US', '{this was made intentionally incorrect}');
if ($mf === null) {
    echo 'Surprise!';
}

?>
```

This inconsistency has been cleaned in PHP 7 : see See Internal Constructor Behavior

See also PHP RFC: Constructor behaviour of internal classes.

9.420.1 Suggestions

- Remove the check on null after a new instantiation

Short name	Classes/NullOnNew
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Instant (5 mins)

9.421 Null Or Boolean Arrays

Null and booleans are valid PHP array base. Yet, they only produces `null` values. They also did not emits any warning until PHP 7.4.

This analysis has been upgraded to cover int and float types too.

```php
<?php

// outputs NULL
```

(continues on next page)

(continued from previous page)

```
var_dump(null[0]);

const MY_CONSTANT = true;
// outputs NULL
var_dump(MY_CONSTANT[10]);

?>
```

See also Null and True.

9.421.1 Suggestions

- Avoid using the array syntax on null and boolean
- Avoid using null and boolean on constant that are expecting arrays

Short name	Arrays/NullBoolean
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.422 Nullable With Constant

Arguments are automatically nullable with a literal null. They used to also be nullable with a constant null, before PHP 8.0.

```
<?php

// Extracted from https://github.com/php/php-src/blob/master/UPGRADING

// Replace
function test(int $arg = CONST_RESOLVING_TO_NULL) {}
// With
function test(?int $arg = CONST_RESOLVING_TO_NULL) {}
// Or
function test(int $arg = null) {}

?>
```

9.422.1 Suggestions

- Use the valid syntax

Short name	Functions/NullableWithConstant
Rulesets	*CompatibilityPHP80*
Php Version	8.0-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

9.423 Nullable Without Check

Nullable typehinted argument should be checked before usage.

```
<?php

// This will emit a fatal error when $a = null
function foo(?A $a) {
    return $a->m();
}

// This is stable
function foo(?A $a) {
    if ($a === null) {
        return 42;
    } else {
        return $a->m();
    }
}

?>
```

9.423.1 Suggestions

-

Short name	Functions/NullableWithoutCheck
Rulesets	*ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.424 Numeric Literal Separator

Integer and floats may be written with internal underscores. This way, it is possible to separate large number into smaller groups, and make them more readable.

Numeric Literal Separators were introduced in PHP 7.4 and are not backward compatible.

```
<?php
$a = 1_000_000_000;   // A billion
$a = 1000000000;      // A billion too...

$b = 107_925_284.88; // 6 light minute to kilometers = 107925284.88 kilometers
$b = 107925284.88;   // Same as above
?>
```

See also PHP RFC: Numeric Literal Separator.

9.424.1 Suggestions

-

Short name	Php/IntegerSeparatorUsage
Rulesets	*CompatibilityPHP73*
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)

9.425 Objects Don't Need References

There is no need to create references for objects, as those are always passed by reference when used as arguments.

Note that when the argument is assigned another value, including another object, then the reference is needed : PHP forgets about reference when they are replaced.

```
<?php

    $object = new stdClass();
    $object->name = 'a';

    foo($object);
    print $object->name; // Name is 'b'

    // No need to make $o a reference
    function foo(&$o) {
        $o->name = 'b';
    }


    // $o is assigned inside the function : it must be called with a &, or the object␣
↪won't make it out of the foo3 scope
    function foo3(&$o) {
        $o = new stdClass;
    }

    $array = array($object);
    foreach($array as &$o) { // No need to make this a reference
        $o->name = 'c';
    }

?>
```

See also Passing by reference.

9.425.1 Suggestions

- Remove the reference
- Assign the argument with a new value

Short name	Structures/ObjectReferences
Rulesets	*Analyze*, *Top10*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
ClearPHP	no-references-on-objects
Examples	*Zencart*, *XOOPS*

9.426 Old Style Constructor

PHP classes used to have the method bearing the same name as the class acts as the constructor. That was PHP 4, and early PHP 5.

The manual issues a warning about this syntax : `Old style constructors are DEPRECATED in PHP 7.0, and will be removed in a future version. You should always use `__construct() <https://www.php.net/manual/en/language.oop5.decon.php>`_ in new code.`

```
<?php

namespace {
    // Global namespace is important
    class foo {
        function foo() {
            // This acts as the old-style constructor, and is reported by PHP
        }
    }

    class bar {
        function __construct() { }
        function bar() {
            // This doesn't act as constructor, as bar has a __construct() method
        }
    }
}

namespace Foo\Bar{
    class foo {
        function foo() {
            // This doesn't act as constructor, as bar is not in the global namespace
        }
    }
}

?>
```

This is no more the case in PHP 5, which relies on `__construct()` to do so. Having this old style constructor may bring in confusion, unless you are also supporting old time PHP 4.

Note that classes with methods bearing the class name, but inside a namespace are not following this convention, as this is not breaking backward compatibility. Those are excluded from the analyze.

See also Constructors and Destructors.

9.426.1 Suggestions

- Remove old style constructor and make it `__construct()`
- Remove old libraries and use a modern component

Short name	Classes/OldStyleConstructor
Rulesets	*Analyze*, *CompatibilityPHP80*
Severity	Minor
Time To Fix	Quick (30 mins)
ClearPHP	no-php4-class-syntax

9.427 Old Style __autoload()

Avoid __autoload(), only use spl_register_autoload().

__autoload() is deprecated since PHP 7.2 and possibly removed in later versions. spl_register_autoload() was introduced in PHP 5.1.0.

__autoload() may only be declared once, and cannot be modified later. This creates potential conflicts between libraries that try to set up their own autoloading schema.

On the other hand, spl_register_autoload() allows registering and de-registering multiple autoloading functions or methods.

```php
<?php

// Modern autoloading.
function myAutoload($class){}
spl_register_autoload('myAutoload');

// Old style autoloading.
function __autoload($class){}

?>
```

Do not use the old __autoload() function, but rather the new spl_register_autoload() function.

See also Autoloading Classe.

9.427.1 Suggestions

- Move to spl_register_autoload()
- Remove usage of the old __autoload() function
- Modernize usage of old libraries

Short name	Php/oldAutoloadUsage
Rulesets	*Analyze*
Severity	Major
Time To Fix	Instant (5 mins)
ClearPHP	use-smart-autoload
Examples	*Piwigo*

9.428 One If Is Sufficient

Nested conditions may be written another way, and reduce the amount of code.

Nested conditions are equivalent to a `&&` condition. As such, they may be switched. When one of the condition has no explicit else, then it is lighter to write it as the first condition. This way, it is written once, and not repeated.

```php
<?php

// Less conditions are written here.
     if($b == 2) {
        if($a == 1) {
             ++$c;
     }
        else {
             ++$d;
     }
    }

// ($b == 2) is double here
    if($a == 1) {
     if($b == 2) {
             ++$c;
     }
    }
    else {
     if($b == 2) {
             ++$d;
     }
    }
?>
```

9.428.1 Suggestions

- Switch the if…then conditions, to reduce the amount of conditions to read.

Short name	Structures/OneIfIsSufficient
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Tikiwiki*

9.429 One Letter Functions

One letter functions seems to be really short for a meaningful name. This may happens for very high usage functions, so as to keep code short, but such functions should be rare.

```php
<?php

// Always use a meaningful name
function addition($a, $b) {
    return $a + $b;
```

(continues on next page)

(continued from previous page)

```
}

// One letter functions are rarely meaningful
function f($a, $b) {
    return $a + $b;
}

?>
```

9.429.1 Suggestions

- Use full names for functions
- Remove the function name altogether : use a closure

Short name	Functions/OneLetterFunctions
Rulesets	*Coding Conventions*, *Semantics*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*ThinkPHP*, *Cleverstyle*

9.430 One Variable String

These strings only contains one variable or property or array.

```
<?php

$a = 0;
$b = "$a"; // This is a one-variable string

// Better way to write the above
$b = (string) $a;

// Alternatives :
$b2 = "$a[1]"; // This is a one-variable string
$b3 = "$a->b"; // This is a one-variable string
$c = "d";
$d = "D";
$b4 = "{$$c}";
$b5 = "{$a->foo()}";

?>
```

When the goal is to convert a variable to a string, it is recommended to use the type casting (string) operator : it is then clearer to understand the conversion. It is also marginally faster, though very little.

See also Strings and Type Juggling.

9.430.1 Suggestions

- Drop the surrounding string, keep the variable (or property. . .)

- Include in the string any concatenation that comes unconditionaly after or before
- Convert the variable to a string with the (type) operator

Short name	Type/OneVariableStrings
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Tikiwiki*, *NextCloud*

9.431 Only Variable For Reference

When a method is requesting an argument to be a reference, it cannot be called with a literal value.

The call must be made with a variable, or any assimilated data container : array, property or static property.

```
<?php

// This is not possible
foo(1,2);

// This is working
foo($a, $b);

function foo($a, &$b) {}

?>
```

Note that PHP may detect this error at linting time, if the method is defined after being called : at that point, PHP will only check the problem during execution. This is definitely the case for methods, compared to functions or static methods.

See also Passing arguments by reference.

9.431.1 Suggestions

- Put the literal value in a variable before calling the method.
- Put the literal value in the default value of the reference argument.

Short name	Functions/OnlyVariableForReference
Rulesets	*Analyze*, *LintButWontExec*
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	Medium

9.432 Only Variable Passed By Reference

When an argument is expected by reference, it is compulsory to provide a container. A container may be a variable, an array, a property or a static property.

This may be linted by PHP, when the function definition is in the same file as the function usage. This is silently linted if definition and usage are separated, if the call is dynamical or made as a method.

```
<?php

function foo(&$bar) { /**/ }

function &bar() { /**/ }

// This is not possible : strtolower() returns a value
foo(strtolower($string));

// This is valid : bar() returns a reference
foo(bar($string));

?>
```

This analysis currently covers functioncalls and static methodcalls, but omits methodcalls.

9.432.1 Suggestions

- Store the previous result in a variable, and then call the function.

Short name	Functions/OnlyVariablePassedByReference
Rulesets	*Analyze*
Severity	Critical
Time To Fix	Slow (1 hour)
Examples	*Dolphin*, *PhpIPAM*

9.433 Only Variable Returned By Reference

Function can't return literals by reference.

When a function returns a reference, it is only possible to return variables, properties or static properties.

Anything else, like literals or static expressions, yield a warning at execution time.

```
<?php

// Can't return a literal number
function &foo() {
    return 3 + rand();
}

// bar must return values that are stored in a
function &bar() {
    $a = 3 + rand();
    return $a;
}

?>
```

Short name	Structures/OnlyVariableReturnedByReference
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)

9.434 Optimize Explode()

Limit explode() results at call time. explode() returns a string, after breaking it into smaller strings, with a delimiter.

By default, explode() breaks the whole string into smaller strings, and returns the array. When not all the elements of the returned array are necessary, using the third argument of explode() speeds up the process, by removing unnecessary work.

```php
<?php

$string = '1,2,3,4,5,';

// explode() returns 2 elements, which are then assigned to the list() call.
list($a, $b) = explode(',', $string, 2);

// explode() returns 6 elements, only two of which are then assigned to the list()␣
↪call. The rest are discarded.
list($a, $b) = explode(',', $string, 2);

// it is not possible to skip the first elements, but it is possible to skip the last␣
↪ones.
echo explode(',', $string, 2)[1];

// This protects PHP, in case $string ends up with a lot of commas
$string = foo(); // usually '1,2' but not known
list($a, $b) = explode(',', $string, 2);
?>
```

Limiting explode() has no effect when the operation is already exact : it simply prevents explode() to cut more than needed if the argument is unexpectedly large.

This optimisation applies to preg_split() and mb_split() too.

This is a micro optimisation, unless the exploded string is large.

9.434.1 Suggestions

- Add a limit to explode() call

Short name	Performances/OptimizeExplode
Rulesets	*Performances*
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

9.435 Or Die

Classic old style failed error management.

```
<?php

// In case the connexion fails, this kills the current script
mysql_connect('localhost', $user, $pass) or die();

?>
```

Interrupting a script will leave the application with a blank page, will make your life miserable for testing. Just don't do that.

See also pg_last_error or PDO::exec.

9.435.1 Suggestions

- Throw an exception
- Trigger an error with trigger_error()
- Use your own error mechanism

Short name	Structures/OrDie
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
ClearPHP	no-implied-if
Examples	*Tine20*, *OpenConf*

9.436 Order Of Declaration

The order used to declare members and methods has a great impact on readability and maintenance. However, practices varies greatly. As usual, being consistent is the most important and useful.

The suggested order is the following : traits, constants, properties, methods. Optional characteristics, like final, static… are not specified. Special methods names are not specified.

```
<?php

class x {
    use traits;

    const CONSTANTS = 1;
    const CONSTANTS2 = 1;
    const CONSTANTS3 = 1;

    private $property = 2;
    private $property2 = 2;
    private $property3 = 2;

    public function foo() {}
```

(continues on next page)

(continued from previous page)

```
    public function foo2() {}
    public function foo3() {}
    public function foo4() {}
}

?>
```

Short name	Classes/OrderOfDeclaration
Rulesets	*Coding Conventions*

9.437 Overwritten Exceptions

In catch blocks, it is good practice to avoid overwriting the incoming exception, as information about the exception will be lost.

```
<?php

try {
    doSomething();
} catch (SomeException $e) {
    // $e is overwritten
    $e = new anotherException($e->getMessage());
    throw $e;
} catch (SomeOtherException $e) {
    // $e is chained with the next exception
    $e = new Exception($e->getMessage(), 0, $e);
    throw $e;
}

?>
```

9.437.1 Suggestions

- Use another variable name to create new values inside the catch
- Use anonymous catch clause (no variable caught) in PHP 8.0, to make this explicit

Short name	Exceptions/OverwriteException
Rulesets	*Analyze*, *Suggestions*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.438 Overwritten Literals

The same variable is assigned a literal twice. It is possible that one of the assignation is too much.

This analysis doesn't take into account the distance between two assignations : it may report false positives when the variable is actually used for several purposes, and, as such, assigned twice with different values.

```php
<?php

function foo() {
    // Two assignations in a short sequence : one is too many.
    $a = 1;
    $a = 2;

    for($i = 0; $i < 10; $i++) {
        $a += $i;
    }
    $b = $a;

    // New assignation. $a is now used as an array.
    $a = array(0);
}

?>
```

Short name	Variables/OverwrittenLiterals
Rulesets	*Analyze*
Severity	Major
Time To Fix	Instant (5 mins)

9.439 Overwritten Source And Value

In a foreach(), it is best to keep source and values distinct. Otherwise, they overwrite each other.

Since PHP 7.0, PHP makes a copy of the original source, then works on it. This makes possible to use the same name for the source and the values.

```php
<?php

// displays 0-1-2-3-3
$array = range(0, 3);
foreach($array as $array) {
    print $array . '-';
}
print_r($array);


/* displays 0-1-2-3-Array
(
    [0] => 0
    [1] => 1
    [2] => 2
    [3] => 3
)
*/
$array = range(0, 3);
foreach($array as $v) {
    print $v . '-';
}
print_r($array);
```

(continues on next page)

(continued from previous page)

```
?>
```

When the source is used as the value, the elements in the array are successively assigned to itself. After the loop, the original array has been replaced by its last element.

The same applies to the index, or to any variable in a list() structure, used in a foreach().

9.439.1 Suggestions

- Keep the source, the index and the values distinct

Short name	Structures/ForeachSourceValue
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*ChurchCRM*, *ExpressionEngine*

9.440 PHP 7.0 New Classes

Those classes are now declared natively in PHP 7.0 and should not be declared in custom code.

There are 8 new classes :

- `Error`
- `ParseError`
- `TypeError`
- `ArithmeticError`
- `DivisionByZeroError`
- `ClosedGeneratorException`
- `ReflectionGenerator`
- `ReflectionType`
- `AssertionError`

```
<?php

namespace {
    // Global namespace
    class Error {
        // Move to a namespace
        // or, remove this class
    }
}

namespace B {
    class Error {
        // This is OK : in a namespace
    }
```

(continues on next page)

(continued from previous page)

```
}

?>
```

See also New Classes and Interfaces.

Short name	Php/Php70NewClasses
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.441 PHP 7.0 New Interfaces

The following interfaces are introduced in PHP 7.0. They shouldn't be defined in custom code.

Short name	Php/Php70NewInterfaces
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.442 PHP 7.0 Removed Directives

List of directives that are removed in PHP 7.0.

Short name	Php/Php70RemovedDirective
Rulesets	*CompatibilityPHP70*, *CompatibilityPHP71*
Php Version	With PHP 7.0 and more recent
Severity	Major
Time To Fix	Slow (1 hour)

9.443 PHP 7.0 Removed Functions

The following PHP native functions were removed in PHP 7.0.

- ereg()
- ereg_replace()
- eregi()
- eregi_replace()
- split()
- spliti()
- sql_regcase()

- magic_quotes_runtime()
- set_magic_quotes_runtime()
- call_user_method()
- call_user_method_array()
- set_socket_blocking()
- mcrypt_ecb()
- mcrypt_cbc()
- mcrypt_cfb()
- mcrypt_ofb()
- datefmt_set_timezone_id()
- imagepsbbox()
- imagepsencodefont()
- imagepsextendfont()
- imagepsfreefont()
- imagepsloadfont()
- imagepsslantfont()
- imagepstext()

This analysis skips redefined PHP functions : when a replacement for a removed PHP function was created, with condition on the PHP version, then its usage is considered valid.

See also PHP 7.0 Removed Functions.

9.443.1 Suggestions

- Replace the old functions with modern functions
- Remove the usage of the old functions
- Create an alternative function by wiring the old name to a new feature

Short name	Php/Php70RemovedFunctions
Rulesets	*CompatibilityPHP70*, *CompatibilityPHP71*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.444 PHP 7.0 Scalar Typehints

New scalar typehints were introduced : `bool`, `int`, `float`, `string`.

They cannot be used before PHP 7.0, and will be confused with classes or interfaces.

```
<?php

function foo(string $name) {
    print Hello $name;
}

foo(Damien);
// display 'Hello Damien'

foo(33);
// displays an error

?>
```

See also Scalar type declarations, and PHP 7 SCALAR TYPE DECLARATIONS.

Short name	Php/PHP70scalartypehints
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and more recent
Severity	Critical
Time To Fix	Quick (30 mins)

9.445 PHP 7.1 Microseconds

PHP supports microseconds in `DateTime` class and date_create() function. This was introduced in PHP 7.1.

In previous PHP versions, those dates only used seconds, leading to lazy comparisons :

```
<?php

$now = date_create();
usleep(10);              // wait for 0.001 ms
var_dump($now == date_create());

?>
```

This code displays true in PHP 7.0 and older, (unless the code was run too close from the next second). In PHP 7.1, this is always false.

This is also true with `DateTime` :

```
<?php

$now = new DateTime();
usleep(10);              // wait for 0.001 ms
var_dump((new DateTime())->format('u') == $now->format('u'));

?>
```

This evolution impacts mostly exact comparisons (== and ===). Non-equality (!= and !==) will probably be always true, and should be reviewed.

See also Backward incompatible changes.

Short name	Php/Php71microseconds
Rulesets	*CompatibilityPHP71*
Severity	Major
Time To Fix	Quick (30 mins)

9.446 PHP 7.1 Removed Directives

List of directives that are removed in PHP 7.1.

Short name	Php/Php71RemovedDirective
Rulesets	*CompatibilityPHP71*
Php Version	With PHP 7.1 and more recent
Severity	Major
Time To Fix	Slow (1 hour)

9.447 PHP 7.1 Scalar Typehints

A new scalar typehint was introduced : iterable.

It can’t be used before PHP 7.1, and will be confused with classes or interfaces.

```
<?php

function foo(iterable $iterable) {
    foreach ($iterable as $value) {
        echo $value.PHP_EOL;
    }
}

foo(range(1,20));
// works with array

foo(new ArrayIterator([1, 2, 3]));
// works with an iterator

foo((function () { yield 1; })() );
// works with a generator

?>
```

See also iterable pseudo-type, and The iterable Pseudo-Type.

Short name	Php/PHP71scalartypehints
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.1 and more recent
Severity	Critical
Time To Fix	Quick (30 mins)

9.448 PHP 7.2 Deprecations

Several functions are deprecated in PHP 7.2.

- parse_str() with no second argument
- assert() on strings
- Usage of gmp_random(), create_function(), each()
- Usage of (unset)
- Usage of `$php_errormsg`
- directive `mbstring.func_overload` (not supported yet)

Deprecated functions and extensions are reported in a separate analysis.

See also Deprecations for PHP 7.2.

9.448.1 Suggestions

- Remove the deprecated functions, and replace them with a new feature
- Use a replacement function to emulate this old behavior

Short name	Php/Php72Deprecation
Rulesets	*CompatibilityPHP72*
Php Version	With PHP 7.2 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.449 PHP 7.2 Object Keyword

'object' is a PHP keyword. It can't be used for class, interface or trait name.

This is the case since PHP 7.2.

```php
<?php

// Valid until PHP 7.2
class object {}

// Altough it is really weird anyway...

?>
```

See also List of Keywords.

Short name	Php/Php72ObjectKeyword
Rulesets	*CompatibilityPHP72*
Php Version	With PHP 7.2 and older
Severity	Major
Time To Fix	Quick (30 mins)

9.450 PHP 7.2 Removed Functions

The following PHP native functions were removed in PHP 7.2.

- png2wbmp()
- jpeg2wbmp()
- create_function()
- gmp_random()
- each()

This analysis skips redefined PHP functions : when a replacement for a removed PHP function was created, with condition on the PHP version, then its usage is considered valid.

See also Deprecated features in PHP 7.2.x.

Short name	Php/Php72RemovedFunctions
Rulesets	*CompatibilityPHP72*
Php Version	With PHP 7.2 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.451 PHP 7.2 Scalar Typehints

A new scalar typehint was introduced : object.

It can't be used before PHP 7.2, and will be confused with classes or interfaces.

```
<?php

function test(object $obj) : object
{
    return new SplQueue();
}

test(new StdClass());

?>
```

See also New object type, and PHP 7.2 and Object Typehint.

Short name	Php/PHP72scalartypehints
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.2 and more recent
Severity	Critical
Time To Fix	Quick (30 mins)

9.452 PHP 7.3 Last Empty Argument

PHP allows the last element of any functioncall to be empty. The argument is then not send.

This was introduced in PHP 7.3, and is not backward compatible.

The last empty line is easier on the VCS, allowing clearer text diffs.

```
<?php

function foo($a, $b) {
    print_r(func_get_args());
}


foo(1,
    2,
    );

foo(1);


?>
```

See also Allow a trailing comma in function calls and Trailing commas.

Short name	Php/PHP73LastEmptyArgument
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.3 and more recent
Severity	Critical
Time To Fix	Quick (30 mins)

9.453 PHP 7.3 Removed Functions

The following PHP native functions were removed in PHP 7.3.

- image2wbmp()

This analysis skips redefined PHP functions : when a replacement for a removed PHP function was created, with condition on the PHP version, then its usage is considered valid.

See also PHP 7.3 Removed Functions.

Short name	Php/Php73RemovedFunctions
Rulesets	*CompatibilityPHP73*
Php Version	With PHP 7.3 and older
Severity	Critical
Time To Fix	Slow (1 hour)

9.454 PHP 7.4 Constant Deprecation

One constant is deprecated in PHP 7.4.

- CURLPIPE_HTTP1

See also Deprecations for PHP 7.2.

9.454.1 Suggestions

- Use CURLPIPE_MULTIPLEX or CURLPIPE_NOTHING

Short name	Php/Php74Deprecation
Rulesets	*CompatibilityPHP74*
Php Version	With PHP 7.4 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.455 PHP 7.4 Removed Directives

List of directives that are removed in PHP 7.4.

- allow_url_include

See Deprecation allow_url_include.

9.455.1 Suggestions

- Stop using this directive

Short name	Php/Php74RemovedDirective
Rulesets	*CompatibilityPHP74*
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)

9.456 PHP 7.4 Removed Functions

The following PHP native functions were deprecated in PHP 7.4.

- hebrevc()
- convert_cyr_string()
- ezmlm_hash()
- money_format()
- restore_include_path()
- get_magic_quotes_gpc()

- get_magic_quotes_runtime()

This analysis skips redefined PHP functions : when a replacement for a removed PHP function was created, with condition on the PHP version, then its usage is considered valid.

See also PHP 7.4 Removed Functions and PHP 7.4 Deprecations : Introduction.

9.456.1 Suggestions

-

Short name	Php/Php74RemovedFunctions
Rulesets	*CompatibilityPHP74*
Php Version	With PHP 7.3 and older
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	Very high

9.457 PHP 7.4 Reserved Keyword

`fn` is a new PHP keyword. In PHP 7.4, it is used to build the arrow functions. When used at an illegal position, `fn` generates a Fatal error at compile time.

As a key word, `fn` is not allowed as constant name, function name, class name or inside namespaces.

```php
<?php

// PHP 7.4 usage of fn
function array_values_from_keys($arr, $keys) {
    return array_map(fn($x) => $arr[$x], $keys);
}

// PHP 7.3 usage of fn
const fn = 1;

function fn() {}

class x {
    // This is valid in PHP 7.3 and 7.4
    function fn() {}
}

?>
```

`fn` is fine for method names. It may also be used for constants with define(), and constant() but it is not recommended.

See also PHP RFC: Arrow Functions.

9.457.1 Suggestions

-

Short name	Php/Php74ReservedKeyword
Rulesets	*CompatibilityPHP74*
Php Version	7.4-
Severity	Minor
Time To Fix	Quick (30 mins)

9.458 PHP 74 New Directives

List of directives that are new in PHP 7.4.

- `zend.exception_ignore_args` : From the php.ini : `Allows to include or exclude arguments from stack traces generated for exceptions. Default: Off`
- `opcache.preload_user`

See RFC Preload.

9.458.1 Suggestions

- Do not use those directives with PHP before version 7.4

Short name	Php/Php74NewDirective
Rulesets	*CompatibilityPHP73*
Php Version	7.4-
Severity	Minor
Time To Fix	Quick (30 mins)

9.459 PHP 8.0 Removed Constants

The following PHP native constants were removed in PHP 8.0.

- INTL_IDNA_VARIANT_2003 (See Deprecate and remove INTL_IDNA_VARIANT_2003)

9.459.1 Suggestions

- Remove usage of INTL_IDNA_VARIANT_2003 and use

Short name	Php/Php80RemovedConstant
Rulesets	*CompatibilityPHP80*
Severity	Critical
Time To Fix	Quick (30 mins)

9.460 PHP 8.0 Removed Directives

List of directives that are removed in PHP 8.0.

In PHP 8.0, *track_errors* was removed.

You can detect valid directives with ini_get(). This native function will return false, when the directive doesn't exist, while actual directive values will be returned as a string.

9.460.1 Suggestions

- Remove usage of *track_errors*.

Short name	Php/Php80RemovedDirective
Rulesets	*CompatibilityPHP80*
Severity	Minor
Time To Fix	Quick (30 mins)

9.461 PHP 8.0 Removed Functions

The following PHP native functions were removed in PHP 8.0.

- image2wbmp()
- png2wbmp()
- jpeg2wbmp()
- ldap_sort()

Short name	Php/Php80RemovedFunctions
Rulesets	*CompatibilityPHP80*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.462 PHP Keywords As Names

PHP has a set of reserved keywords. It is recommended not to use those keywords for names structures.

PHP does check that a number of structures, such as classes, methods, interfaces… can't be named or called using one of the keywords. However, in a few other situations, no check are enforced. Using keywords in such situation is confusing.

```
<?php

// This keyword is reserved since PHP 7.2
class object {
    // _POST is used by PHP for the $_POST variable
    // This methods name is probably confusing,
    // and may attract more than its share of attention
    function _POST() {

    }
}

?>
```

See also List of Keywords, Predefined Classes, Predefined Constants, List of other reserved words and Predefined Variables.

9.462.1 Suggestions

- Rename the structure
- Choose another naming convention to avoid conflict and rename the current structures

Name	Default	Type	Description
reservedNames		string	Other reserved names : all in a string, comma separated.
allowedNames		string	PHP reserved names that can be used in the code. All in a string, comma separated.

Short name	Php/ReservedNames
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*ChurchCRM*, *xataface*

9.463 PHP5 Indirect Variable Expression

Indirect variable expressions changes between PHP 5 an 7.

The following structures are evaluated differently in PHP 5 and 7. It is recommended to review them or switch to a less ambiguous syntax.

```
<?php

// PHP 7
$foo = 'bar';
$bar['bar']['baz'] = 'foobarbarbaz';
echo $$foo['bar']['baz'];
echo ($$foo)['bar']['baz'];

// PHP 5
$foo['bar']['baz'] = 'bar';
$bar = 'foobarbazbar';
echo $$foo['bar']['baz'];
echo ${$foo['bar']['baz']};

?>
```

See Backward incompatible changes PHP 7.0

Expression	PHP 5 interpretation	PHP 7 interpretation
$$foo[‘bar’][‘baz’] $foo->$bar[‘baz’] $foo->$bar[‘baz’]() Foo::$bar[‘baz’]()	${$foo[‘bar’][‘baz’]} $foo->{$bar[‘baz’]} $foo->{$bar[‘baz’]}() Foo::{$bar[‘baz’]}()	($$foo)[‘bar’][‘baz’] ($foo->$bar)[‘baz’] ($foo->$bar)[‘baz’]() (Foo::$bar)[‘baz’]()

9.463.1 Suggestions

- Avoid using complex expressions, mixing `$$\`, `[0]` and `->` in the same expression
- Add curly braces {} to ensure that the precedence is the same between PHP 5 and 7. For example, `$$v` becomes `${$v}`

Short name	Variables/Php5IndirectExpression
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.464 PHP7 Dirname

With PHP 7, dirname() has a second argument that represents the number of parent folder to follow. This prevent us from using nested dirname() calls to reach an grand-parent direct.

```
<?php
$path = '/a/b/c/d/e/f';

// PHP 7 syntax
$threeFoldersUp = dirname($path, 3);

// PHP 5 syntax
$threeFoldersUp = dirname(dirname(dirname($path)));

?>
```

See also dirname.

9.464.1 Suggestions

- Use dirname()'s second argument

Short name	Structures/PHP7Dirname
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *Suggestions*, *php-cs-fixable*
Php Version	7.0+
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*OpenConf*, *MediaWiki*

9.465 Parameter Hiding

When a parameter is set to another variable, and never used.

While this is a legit syntax, parameter hiding tends to make the code confusing. The parameter itself seems to be unused, while some extra variable appears.

Keep this code simple by removing the hiding parameter.

```
<?php

function substract($a, $b) {
    // $b is given to $c;
    $c = $b;

    $c is used, but $b would be the same
    return $a - $c;
}

?>
```

9.465.1 Suggestions

- Remove the hiding parameter

Short name	Functions/ParameterHiding
Rulesets	*Semantics*
Severity	Minor
Time To Fix	Quick (30 mins)

9.466 Parent First

When calling parent constructor, always put it first in the `__construct` method. It ensures the parent is correctly build before the child start using values.

```
<?php

class father {
    protected $name = null;

    function __construct() {
        $this->name = init();
    }
}

class goodSon {
    function __construct() {
        // parent is build immediately,
        parent::__construct();
        echo my name is.$this->name;
    }
}

class badSon {
    function __construct() {
        // This will fail.
        echo my name is.$this->name;
```

(continues on next page)

(continued from previous page)

```
        // parent is build later,
        parent::__construct();
    }
}

?>
```

This analysis doesn’t apply to Exceptions.

9.466.1 Suggestions

- Use `parent\:\:__construct` as the first call in the constructor.

Short name	Classes/ParentFirst
Rulesets	*Analyze*, *Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*shopware*, *PrestaShop*

9.467 Parent, Static Or Self Outside Class

Parent, static and self keywords must be used within a class or a trait. They make no sens outside a class or trait scope, as self and static refers to the current class and parent refers to one of parent above.

PHP 7.0 and later detect their usage at compile time, and emits a fatal error.

```
<?php

class x {
    const Y = 1;

    function foo() {
        // self is \x
        echo self::Y;
    }
}

const Z = 1;
// This lint but won't anymore
echo self::Z;

?>
```

Static may be used in a function or a closure, but not globally.

Short name	Classes/PssWithoutClass
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)

9.468 Parenthesis As Parameter

Using parenthesis around parameters used to silent some internal check. This is not the case anymore in PHP 7, and should be fixed by removing the parenthesis and making the value a real reference.

```
<?php

// PHP 7 sees through parenthesis
$d = foo(1, 2, $c);

// Avoid parenthesis in arguments
$d = foo(1, 2, ($c));

?>
```

9.468.1 Suggestions

- Remove the parenthesis when they are only encapsulating an argument

Short name	Php/ParenthesisAsParameter
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Severity	Major
Time To Fix	Quick (30 mins)

9.469 Pathinfo() Returns May Vary

pathinfo() function returns an array whose content may vary. It is recommended to collect the values after check, rather than directly.

```
<?php

$file = '/a/b/.c';
//$extension may be missing, leading to empty $filename and filename in $extension
list( $dirname, $basename, $extension, $filename ) = array_values( pathinfo($file) );

//Use PHP 7.1 list() syntax to assign correctly the values, and skip array_values()
//This emits a warning in case of missing index
['dirname'   => $dirname,
 'basename'  => $basename,
 'extension' => $extension,
 'filename'  => $filename ] = pathinfo($file);

//This works without warning
$details = pathinfo($file);
$dirname   = $details['dirname'] ?? getpwd();
$basename  = $details['basename'] ?? '';
$extension = $details['extension'] ?? '';
$filename  = $details['filename'] ?? '';

?>
```

The same applies to parse_url(), which returns an array with various index.

9.469.1 Suggestions

- Add a check on the return value of pathinfo() before using it.

Short name	Php/PathinfoReturns
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*NextCloud*

9.470 Php 7 Indirect Expression

Those are variable indirect expressions that are interpreted differently in PHP 5 and PHP 7.

You should check them so they don't behave strangely.

```php
<?php

// Ambiguous expression :
$b = $$foo['bar']['baz'];
echo $b;

$foo = array('bar' => array('baz' => 'bat'));
$bat = 'PHP 5.6';

// In PHP 5, the expression above means :
$b = $\{$foo['bar']['baz']};
$b = 'PHP 5.6';

$foo = 'a';
$a = array('bar' => array('baz' => 'bat'));

// In PHP 7, the expression above means :
$b = ($$foo)['bar']['baz'];
$b = 'bat';

?>
```

See also Changes to variable handling.

9.470.1 Suggestions

- Avoid using complex expressions, mixing $$, [0] and -> in the same expression
- Add curly braces { } to ensure that the precedence is the same between PHP 5 and 7. For example, `$$v` becomes `${$v}`

Short name	Variables/Php7IndirectExpression
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*
Php Version	With PHP 7.0 and more recent
Severity	Major
Time To Fix	Slow (1 hour)

9.471 Php 7.1 New Class

New classes, introduced in PHP 7.1. If classes where created with the same name, in current code, they have to be moved in a namespace, or removed from code to migrate safely to PHP 7.1.

The new class is : ReflectionClassConstant. The other class is 'Void' : this is forbidden as a class name, as Void is used for return type hint.

```
<?php

class ReflectionClassConstant {
    // Move to a namespace, do not leave in global
    // or, remove this class
}

?>
```

Short name	Php/Php71NewClasses
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.1 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.472 Php 7.2 New Class

New classes, introduced in PHP 7.2. If classes where created with the same name, in current code, they have to be moved in a namespace, or removed from code to migrate safely to PHP 7.2.

The new class is : HashContext.

```
<?php

namespace {
    // Global namespace
    class HashContext {
        // Move to a namespace
```

(continues on next page)

(continued from previous page)

```
        // or, remove this class
    }
}

namespace B {
    class HashContext {
        // This is OK : in a namespace
    }
}

?>
```

Short name	Php/Php72NewClasses
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP72*
Php Version	With PHP 7.2 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.473 Php 7.4 New Class

New classes, introduced in PHP 7.4. If classes where created with the same name, in current code, they have to be moved in a namespace, or removed from code to migrate safely to PHP 7.4.

The new classes are :

- `ReflectionReference`
- `WeakReference`

```
<?php

namespace {
    // Global namespace
    class WeakReference {
        // Move to a namespace
        // or, remove this class
    }
}

namespace B {
    class WeakReference {
        // This is OK : in a namespace
    }
}

?>
```

9.473.1 Suggestions

- Move the current classes with the same names into a distinct domain name

Short name	Php/Php74NewClasses
Rulesets	*CompatibilityPHP74*
Php Version	With PHP 7.2 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.474 Php 8.0 Only TypeHints

Two scalar typehints are introduced in version 8. They are `false` and `null`. In PHP 7.0, both those values could not be used as a class or interface name, to avoid confusion with the actual booleans, nor `null` value.

`false` represents a false boolean, and nothing else. It is more restrictive than a boolean, which accepts true too. `null` is an alternative syntax to `?` : it allows the type to be `null`.

Both the above typehints are to be used in cunjunction with other types : they can't be used alone.

```
<?php

// function accepts an A object, or null.
function foo(A|null $x) {}

// same as above
function foo2(A|null $x) {}

// returns an object of class B, or false
function bar($x) : false|B {}

?>
```

See also PHP RFC: Union Types 2.0.

9.474.1 Suggestions

-

Short name	Php/Php80OnlyTypeHints
Rulesets	*CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*, *CompatibilityPHP74*, *CompatibilityPHP56*
Severity	Minor
Time To Fix	Quick (30 mins)

9.475 Php 8.0 Variable Syntax Tweaks

Several variable syntaxes are added in version 8.0. They extends the PHP 7.0 syntax updates, and fix a number of edges cases.

In particular, new``and ``instanceof now support a way to inline the expression, rather than use a temporary variable.

Magic constants are now accessible with array notation, just like another constant. It is also possible to use method calls : although this is Syntacticly correct for PHP, this won't be executed, as the left operand is a string, and not an object.

```php
<?php

 // array name is dynamically build
 echo foo$bar[0];
 // static method
 foo$bar::baz();
 // static property
 foo$bar::$baz;

 // Syntactly correct, but not executable
 foo$bar->baz();

 // expressions with instanceof and new
    $object = new (class_.$name);
    $x instanceof (class_$name);

    // PHP 7.0 style
    $className = class_.$name;
    $object = new $className;

?>
```

See also PHP RFC: Variable Syntax Tweaks and scalar_objects in PHP.

Short name	Php/Php80VariableSyntax
Rulesets	*CompatibilityPHP74*
Php Version	8.0+
Severity	Major
Time To Fix	Quick (30 mins)

9.476 Php/UseMatch

9.476.1 Suggestions

-

Short name	Php/UseMatch
Rulesets	*CompatibilityPHP74*
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)

9.477 Php7 Relaxed Keyword

Most of the traditional PHP keywords may be used inside classes, trait or interfaces.

```
<?php

// Compatible with PHP 7.0 +
class foo {
    // as is a PHP 5 keyword
    public function as() {

    }
}

?>
```

This was not the case in PHP 5, and will yield parse errors.

See also Loosening Reserved Word Restrictions.

Short name	Php/Php7RelaxedKeyword
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and more recent
Severity	Major
Time To Fix	Slow (1 hour)

9.478 Phpinfo

phpinfo() is a great function to learn about the current configuration of the server.

```
<?php

if (DEBUG) {
    phpinfo();
}

?>
```

If left in the production code, it may lead to a critical leak, as any attacker gaining access to this data will know a lot about the server configuration.

It is advised to never leave that kind of instruction in a production code.

phpinfo() may be necessary to access some specific configuration of the server : for example, `Apache` module list are only available via phpinfo(), and apache_get(), when they are loaded.

9.478.1 Suggestions

- Remove all usage of phpinfo()
- Add one or more constant to fine-tune the phpinfo(), and limit the amount of displayed information
- Replace phpinfo() with a more adapted method : get_loaded_extensions() to access the list of loaded extensions

Short name	Structures/PhpinfoUsage
Rulesets	*Security*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Dolphin*

9.479 Possible Alias Confusion

An alias is used for a class that doesn't belong to the current namespace, while there is such a class. This also applies to traits and interfaces.

When no alias is used, PHP will search for a class in the local space. Since classes, traits and interfaces are usually stored one per file, it is a valid syntax to create an alias, even if this alias name is the name of a class in the same namespace.

Yet, with an alias refering to a remote class, while a local one is available, it is possible to generate confusion.

```
<?php

// This should be in a separate file, but has been merged here, for display purposes.
namespace A {
    //an alias from a namespace called C
    use C\A as C_A;

    //an alias from a namespace called C, which will superseed the local A\B class␣
↪(see below)
    use C\D as B;
}

namespace A {
    // There is a class B in the A namespace
    class B {}
}

?>
```

9.479.1 Suggestions

- Avoid using existing classes names for alias
- Use a coding convention to distinguish alias from names

Short name	Namespaces/AliasConfusion
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.480 Possible Increment

This expression looks like a typo : a missing + would change the behavior.

The same pattern is not reported with -, as it is legit expression. + sign is usually understated, rather than explicit.

```
<?php

// could it be a ++$b ?
$a = +$b;

?>
```

See also Incrementing/Decrementing Operators and Arithmetic Operators.

9.480.1 Suggestions

- Drop the whole assignation
- Complete the addition with another value : $a = 1 + $b
- Make this a ++ operator : ++$b
- Make this a negative operator : -$b
- Make the casting explicit : (int) $b

Short name	Structures/PossibleIncrement
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Zurmo*, *MediaWiki*

9.481 Possible Infinite Loop

Loops on files that can't be open results in infinite loop.

fgets(), and functions like fgetss(), fgetcsv(), fread(), return false when they finish reading, or can't access the file.

In case the file is not accessible, comparing the result of the reading to something that is falsy, leads to a permanent valid condition. The execution will only finish when the `max_execution_time` is reached.

```
<?php

$file = fopen('/path/to/file.txt', 'r');
// when fopen() fails, the next loops is infinite
// fgets() will always return false, and while will always be true.
while($line = fgets($file) != 'a') {
    doSomething();
}

?>
```

It is recommended to check the file resources when they are opened, and always use === or !== to compare readings. feof() is also a reliable function here.

Short name	Structures/PossibleInfiniteLoop
Rulesets	*Analyze*
Severity	Critical
Time To Fix	Quick (30 mins)

9.482 Possible Missing Subpattern

When capturing subpatterns are the last ones in a regex, PHP doesn't fill their spot in the resulting array. This leads to a possible missing index in the result array.

```
<?php

// displays a partial array, from 0 to 1
preg_match('/(a)(b)?/', 'adc', $r);
print_r($r);
/*
Array
(
    [0] => a
    [1] => a
)
*/

// displays a full array, from 0 to 2
preg_match('/(a)(b)?/', 'abc', $r);
print_r($r);

/*
Array
(
    [0] => ab
    [1] => a
    [2] => b
)
*/

// double 'b' when it is found
print preg_replace(',^a(b)?,', './$1$1', 'abc'); // prints ./abbc
print preg_replace(',^a(b)?,', './$1$1', 'adc'); // prints ./dc

?>
```

?>

The same applies to preg_replace() : the pattern may match the string, but no value is available is the corresponding sub-pattern.

In PHP 7.4, a new option was added : PREG_UNMATCHED_AS_NULL, which always provides a value for the subpatterns.

See also Bug #50887 preg_match , last optional sub-patterns ignored when empty and Bug #73948 Preg_match_all should return NULLs on trailing optional capture groups..

9.482.1 Suggestions

- Add an always capturing subpatterns after the last ?
- Move the ? inside the parenthesis, so the parenthesis is always on, but the content may be empty
- Add a test on the last index of the resulting array, to ensure it is available when needed
- Use the PREG_UNMATCHED_AS_NULL option (PHP 7.4+)

Short name	Php/MissingSubpattern
Rulesets	*Analyze*, *Top10*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*phpMyAdmin*, *SPIP*

9.483 Pre-increment

When possible, use the pre-increment operator (`++$i` or `--$i`) instead of the post-increment operator (`$i++` or `$i--`).

The latter needs an extra memory allocation that costs about 10% of performances.

```
<?php

// ++$i should be preferred over $i++, as current value is not important
for($i = 0; $i <10; ++$i) {
    // do Something
}

// ++$b and $b++ have different impact here, since $a will collect $b + 1 or $b,␣
↪respectively.
$a = $b++;

?>
```

This is a micro-optimisation. However, its usage is so widespread, including within loops, that it may eventually have an significant impact on execution time. As such, it is recommended to adopt this rule, and only consider changing legacy code as they are refactored for other reasons.

9.483.1 Suggestions

- Use the pre increment when the new value is not reused.

Short name	Performances/PrePostIncrement
Rulesets	*Analyze*, *Performances*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*ExpressionEngine*, *Traq*

9.484 Prefix And Suffixes With Typehint

This analysis checks the relationship between methods prefixes and suffixes, with their corresponding return typehint.

For example, a method with the signature `function isACustomer() {}` should return a boolean. That boolean can then be read when calling the method : `if ($user->isACustomer()) {}`.

There are multiple such convention that may be applied. For example, `has*` should return a boolean, `set*` should return nothing (a.k.a `void`), and ‘‘get*‘‘shall return any kind of type.

```php
<?php

class x  {
    // Easy to read convention
    function isAUser() : bool {}

    // shall return a boolean
    function isACustomer() {}

    // shall return a string, based on suffix 'name => string'
    function getName() {}

    // shall return a string, based on suffix 'name => string'
    function getUsername() {}

    // shall return \Uuid, based on prefix 'uuid => \Uuid'
    function getUuid() {}

    // shall return anything, based on no prefix nor suffix
    function getBirthday() {}

}

?>
```

There are 2 parameters for this analysis. It is recommended to customize them to get an better results, related to the naming conventions used in the code.

`prefixedType` is used for prefix in method names, which is the beginning of the name. `suffixedType` is used for suffixes : the ending part of the name. Matching is case insensitive.

The prefix is configured as the index of the map, while the related type is configured as the value of the map.

`prefixToType['is'] = 'bool';` will be use as `is*` shall use the `bool` typehint.

Multiple typehints may be used at the same time. PHP supports multiple types since PHP 8.0, and Exakat will support them with any PHP version. Specify multiple types by separating them with comma. Any typehint not found in this list will be reported, including `null`.

PHP scalar types are available : `string`, `int`, `void`, etc. Explicit types, based on classes or interfaces, must use the fully qualified name, not the short name. `suffixToType['uuid'] = '\Uuid';` will be use as `*uuid` shall use the `\Uuid` typehint.

When multiple rules applies, only one is reported.

9.484.1 Suggestions

-

Name	Default	Type	Description
prefixedType	prefixedType['is'] = 'bool'; prefixedType['has'] = 'bool'; prefixedType['set'] = 'void'; prefixedType['list'] = 'array';	ini_hash	List of prefixes and their expected return-type
suffixedType	prefixedType['list'] = 'bool'; prefixedType['int'] = 'int'; prefixedType['string'] = 'string'; prefixedType['name'] = 'string'; prefixedType['description'] = 'string'; prefixedType['id'] = 'int'; prefixedType['uuid'] = 'Uuid';	ini_hash	List of suffixes and their expected return-type

Short name	Functions/PrefixToType
Rulesets	*Semantics*
Severity	Minor
Time To Fix	Quick (30 mins)

9.485 Preprocess Arrays

Using long list of assignations for initializing arrays is significantly slower than the declaring them as an array.

```php
<?php

// Slow way
$a = []; // also with $a = array();
$a[1] = 2;
$a[2] = 3;
$a[3] = 5;
$a[4] = 7;
$a[5] = 11;

// Faster way
$a = [1 => 2,
      2 => 3,
      3 => 5,
      4 => 7,
      5 => 11];

// Even faster way if indexing is implicit
$a = [2, 3, 5, 7, 11];

?>
```

If the array has to be completed rather than created, it is also faster to use += when there are more than ten elements to add.

```php
<?php

// Slow way
$a = []; // also with $a = array();
$a[1] = 2;
$a[2] = 3;
$a[3] = 5;
// some expressions to get $seven and $eleven
```

(continues on next page)

(continued from previous page)

```
$a[4] = $seven;
$a[5] = $eleven;

// Faster way
$a = [1 => 2,
      2 => 3,
      3 => 5];
// some expressions to get $seven and $eleven
$a += [4 => $seven,
       5 => $eleven];

// Even faster way if indexing is implicit
$a = [2, 3, 5];
// some expressions to get $seven and $eleven
$a += [$seven, $eleven];

?>
```

9.485.1 Suggestions

- Preprocess the code so PHP doesn't do it. Keep the detailed version into comments.

Short name	Arrays/ShouldPreprocess
Rulesets	none
Severity	Minor
Time To Fix	Quick (30 mins)

9.486 Preprocessable

The following expression are made of literals or already known values : they may be fully calculated before running PHP.

```
<?php

// Building an array from a string
$name = 'PHP'.' '.'7.2';

// Building an array from a string
$list = explode(',', 'a,b,c,d,e,f');

// Calculating a power
$kbytes = $bytes / pow(2, 10);

// This will never change
$name = ucfirst(strtolower('PARIS'));

?>
```

By doing so, this will reduce the amount of work of PHP.

9.486.1 Suggestions

- Do the work yourself, instead of giving it to PHP

Short name	Structures/ShouldPreprocess
Rulesets	*Analyze*, *Suggestions*, *Suggestions*, *Rector*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*phpadsnew*

9.487 Print And Die

Die() <https://www.php.net/'die>'_ also prints.

When stopping a script with die() <https://www.php.net/'die>'_, it is possible to provide a message as first argument, that will be displayed at execution. There is no need to make a specific call to print or echo.

```
<?php

//  die may do both print and die.
echo 'Error message';
die();

//  exit may do both print and die.
print 'Error message';
exit;

//  exit cannot print integers only : they will be used as status report to the␣
↪system.
print 'Error message';
exit 1;

?>
```

Short name	Structures/PrintAndDie
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)

9.488 Printf Number Of Arguments

The number of arguments provided to printf() or vprintf() doesn't match the format string.

Extra arguments are ignored, and are dead code as such. Missing arguments are reported with a warning, and nothing is displayed.

Omitted arguments produce an error.

```
<?php

// not enough
```

(continues on next page)

(continued from previous page)

```
printf(' a %s ', $a1);
// OK
printf(' a %s ', $a1, $a2);
// too many
printf(' a %s ', $a1, $a2, $a3);

// not enough
sprintf(' a %s ', $a1);
// OK
\sprintf(' a %s ', $a1, $a2);
// too many
sprintf(' a %s ', $a1, $a2, $a3);

?>
```

See also printf and sprintf.

Short name	Structures/PrintfArguments
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*PhpIPAM*

9.489 Processing Collector

When accumulating data in a variable, within a loop, it is slow to apply repeatedly a function to the variable.

The example below illustrate the problem : `$collector` is build with element from `$array`. `$collector` actually gets larger and larger, slowing the in_array() call each time.

It is better to apply the preg_replace() to `$a`, a short variable, and then, add `$a` to the collector.

```
<?php

// Fast way
$collector = '';
foreach($array as $a){
    $a = preg_replace('/__(.*?)__/', '<b>$1</b>', $a);
    $collector .= $a;
}

// Slow way
$collector = '';
foreach($array as $a){
    $collector .= $a;
    $collector = preg_replace('/__(.*?)__/', '<b>$1</b>', $collector);
}

?>
```

9.489.1 Suggestions

- Avoid applying the checks on the whole data, rather on the diff only.

Short name	Performances/RegexOnCollector
Rulesets	*Performances*
Severity	Minor
Time To Fix	Slow (1 hour)

9.490 Property Could Be Local

A property only used in one method may be turned into a local variable.

Public an protected properties are omitted here : they may be modified somewhere else, in the code. This analysis may be upgraded to support those properties, when tracking of such properties becomes available.

Classes where only one non-magic method is available are omitted.

Traits with private properties are processed the same way.

```php
<?php

class x {
    private $foo = 1;

    // Magic method, and constructor in particular, are omitted.
    function __construct($foo) {
        $this->foo = $foo;
    }

    function bar() {
        $this->foo++;

        return $this->foo;
    }

    function barbar() {}
}

?>
```

9.490.1 Suggestions

- Remove the property and make it an argument in the method
- Use that property elsewhere

Short name	Classes/PropertyCouldBeLocal
Rulesets	*Analyze*, *ClassReview*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*Mautic*, *Typo3*

9.491 Property Could Be Private Property

The following properties are never used outside their class of definition Given the analyzed code, they could be set as private.

```php
<?php

class foo {
    public $couldBePrivate = 1;
    public $cantdBePrivate = 1;

    function bar() {
        // couldBePrivate is used internally.
        $this->couldBePrivate = 3;
    }
}

class foo2 extends foo {
    function bar2() {
        // cantdBePrivate is used in a child class.
        $this->cantdBePrivate = 3;
    }
}

//$couldBePrivate is not used outside
$foo = new foo();

//$cantdBePrivate is used outside the class
$foo->cantdBePrivate = 2;

?>
```

Note that dynamic properties (such as $x->$y) are not taken into account.

9.491.1 Suggestions

- Remove the unused property
- Use the private property
- Change the visibility to allow access the property from other part of the code

Short name	Classes/CouldBePrivate
Rulesets	*ClassReview*
Severity	Minor
Time To Fix	Slow (1 hour)

9.492 Property Used In One Method Only

Properties should be used in several methods. When a property is used in only one method, this should have be of another shape.

Properties used in one method only may be used several times, and read only. This may be a class constant. Such properties are meant to be overwritten by an extending class, and that's possible with class constants.

Properties that read and written may be converted into a variable, static to the method. This way, they are kept close to the method, and do not pollute the object's properties.

```
<?php

class foo {
    private $once = 1;
    const ONCE = 1;
    private $counter = 0;

    function bar() {
        // $this->once is never used anywhere else.
        someFunction($this->once);
        someFunction(self::ONCE);   // Make clear that it is a
    }

    function bar2() {
        static $localCounter = 0;
        $this->counter++;

        // $this->once is only used here, for distinguising calls to someFunction2
        if ($this->counter > 10) { // $this->counter is used only in bar2, but it may␣
↪be used several times
            return false;
        }
        someFunction2($this->counter);

        // $localCounter keeps track for all the calls
        if ($localCounter > 10) {
            return false;
        }
        someFunction2($localCounter);
    }
}

?>
```

Note : properties used only once are not returned by this analysis. They are omitted, and are available in the analysis *Used Once Property*.

9.492.1 Suggestions

- Drop the property, and inline the value
- Drop the property, and make the property a local variable
- Use the property in another method

Short name	Classes/PropertyUsedInOneMethodOnly
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*Contao*

9.493 Property Variable Confusion

Within a class, there is both a property and variables bearing the same name.

```
<?php
class Object {
    private $x;

    function SetData( ) {
        $this->x = $x + 2;
    }
}
?>
```

The property and the variable may easily be confused one for another and lead to a bug.

Sometimes, when the property is going to be replaced by the incoming argument, or data based on that argument, this naming schema is made on purpose, indicating that the current argument will eventually end up in the property. When the argument has the same name as the property, no warning is reported.

9.493.1 Suggestions

- Use different names for the properties and variables
- Adopt and apply a naming convention for variables and properties.

Short name	Structures/PropertyVariableConfusion
Rulesets	*Semantics*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*PhpIPAM*

9.494 Queries In Loops

Avoid querying databases in a loop.

Querying an external database in a loop usually leads to performances problems. This is also called the ‘n + 1 problem’.

This problem applies also to prepared statement : when such statement are called in a loop, they are slower than one-time large queries.

It is recommended to reduce the number of queries by making one query, and dispatching the results afterwards. This is true with SQL databases, graph queries, LDAP queries, etc.

```
<?php

// Typical N = 1 problem : there will be as many queries as there are elements in
↪$array
$ids = array(1,2,3,5,6,10);

$db = new SQLite3('mysqlitedb.db');

// all the IDS are merged into the query at once
$results = $db->query('SELECT bar FROM foo WHERE id  in ('.implode(',', $id).')');
```

(continues on next page)

(continued from previous page)

```
while ($row = $results->fetchArray()) {
    var_dump($row);
}


// Typical N = 1 problem : there will be as many queries as there are elements in
↪$array
$ids = array(1,2,3,5,6,10);

$db = new SQLite3('mysqlitedb.db');

foreach($ids as $id) {
    $results = $db->query('SELECT bar FROM foo WHERE id = '.$id);
    while ($row = $results->fetchArray()) {
        var_dump($row);
    }
}

?>
```

This optimisation is not always possible : for example, some SQL queries may not be prepared, like `DROP TABLE` or `DESC`. `UPDATE` commands often update one row at a time, and grouping such queries may be counter-productive or unsafe.

9.494.1 Suggestions

- Batch calls by using WHERE clauses and applying the same operation to all similar data
- Use native commands to avoid double query : REPLACE instead of SELECT-(UPDATE/INSERT), or UPSERT, for example

Short name	Structures/QueriesInLoop
Rulesets	*Analyze*, *Top10*
Severity	Major
Time To Fix	Slow (1 hour)
Examples	*TeamPass*, *OpenEMR*

9.495 Raised Access Level

A property's visibility may be lowered, but not raised.

This error may be detected when the classes are all in the same file : then, PHP reports the problem. However, when the classes are separated in different files, as it is customary, PHP won't check this at linting time, yielding a fatal error at execution time.

First file.

```
<?php

class Foo {
    public $publicProperty;
    protected $protectedProperty;
```

(continues on next page)

(continued from previous page)

```
    private $privateProperty;
}
?>
```

Second file.

```
<?php

class Bar extends Foo {
    private $publicProperty;
    private $protectedProperty;
    private $privateProperty;   // This one is OK
}
?>
```

See also Visibility and Understanding the concept of visibility in object oriented php.

9.495.1 Suggestions

- Lower the visibility in the child class
- Raise the visibility in the parent class

Short name	Classes/RaisedAccessLevel
Rulesets	*ClassReview*, *LintButWontExec*
Severity	Critical
Time To Fix	Quick (30 mins)

9.496 Random Without Try

random_int() and random_bytes() require a try/catch structure around them.

random_int() and random_bytes() emit Exceptions if they meet a problem. This way, failure can't be mistaken with returning an empty value, which leads to lower security.

```
<?php

try {
    $salt = random_bytes($length);
} catch (TypeError $e) {
    // Error while reading the provided parameter
} catch (Exception $e) {
    // Insufficient random data generated
} catch (Error $e) {
    // Error with the provided parameter : <= 0
}

?>
```

Since PHP 7.4, openssl_random_pseudo_bytes() has adopted the same behavior. It is included in this analysis : check your PHP version for actual application.

9.496.1 Suggestions

- Add a try/catch structure around calls to random_int() and random_bytes().

Short name	Structures/RandomWithoutTry
Rulesets	*Security*
Php Version	With PHP 7.0 and more recent
Severity	Critical
Time To Fix	Quick (30 mins)

9.497 Randomly Sorted Arrays

Those literal arrays are written in several places, but their items are in various orders.

This may reduce the reading and proofing of the arrays, and induce confusion. The random order may also be a residue of development : both arrays started with different values, but they grew overtime to handle the same items. The way they were written lead to the current order.

Unless order is important, it is recommended to always use the same order when defining literal arrays. This makes it easier to match different part of the code by recognizing one of its literal.

```
<?php

// an array
$set = [1,3,5,9,10];

function foo() {
    // an array, with the same values but different order, in a different context
    $list = [1,3,5,10,9,];
}

// an array, with the same order than the initial one
$inits = [1,3,5,9,10];

?>
```

9.497.1 Suggestions

- Match the sorting order of the arrays. Choose any of them.
- Configure a constant and use it as a replacement for those arrays.
- Leave the arrays intact : the order may be important.
- For hash arrays, consider turning the array in a class.

Short name	Arrays/RandomlySortedLiterals
Rulesets	*Analyze*, *Suggestions*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*Contao*, *Vanilla*

9.498 Redeclared PHP Functions

Function that bear the same name as a PHP function, and that are declared.

This is useful when managing backward compatibility, like emulating an old function, or preparing for newer PHP versions, like emulating new upcoming function.

```php
<?php

if (version_compare(PHP_VERSION, 7.0) > 0) {
    function split($separator, $string) {
        return explode($separator, $string);
    }
}

print_r( split(' ', '2 3'));

?>
```

9.498.1 Suggestions

- Check if it is still worth emulating that function

Short name	Functions/RedeclaredPhpFunction
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Slow (1 hour)

9.499 Redefined Class Constants

Redefined class constants.

Class constants may be redefined, though it is prone to errors when using them, as it is now crucial to use the right class name to access the right value.

```php
<?php

class a {
    const A = 1;
}

class b extends a {
    const A = 2;
}

class c extends c { }

echo a::A, ' ', b::A, ' ', c::A;
// 1 2 2

?>
```

It is recommended to use distinct names.

Short name	Classes/RedefinedConstants
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Slow (1 hour)

9.500 Redefined Default

Classes allows properties to be set with a default value. When those properties get, unconditionally, another value at constructor time, then one of the default value are useless. One of those definition should go : it is better to define properties outside the constructor.

```php
<?php

class foo {
    public $redefined = 1;

    public function __construct( ) {
        $this->redefined = 2;
    }
}

?>
```

9.500.1 Suggestions

- Move the default assignation to the property definition
- Drop the reassignation in the constructor

Short name	Classes/RedefinedDefault
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Slow (1 hour)
Examples	*Piwigo*

9.501 Redefined Private Property

Private properties are local to their defined class. PHP doesn't forbid the re-declaration of a private property in a child class.

However, having two or more properties with the same name, in the class hierarchy tends to be error prone.

```php
<?php

class A {
    private $isReady = true;
}

class B {
    private $isReady = false;
```

(continues on next page)

(continued from previous page)

```
}

?>
```

Short name	Classes/RedefinedPrivateProperty
Rulesets	*Analyze*
Severity	Major
Time To Fix	Slow (1 hour)
Examples	*Zurmo*

9.502 Redefined Property

Property redefined in a parent class.

Using heritage, it is possible to define several times the same property, at different levels of the hierarchy.

```
<?php

class foo {
    protected $aProperty = 1;
}

class bar extends foo {
    // This property is redefined in the parent class, leading to potential confusion
    protected $aProperty = 1;
}

?>
```

When this is the case, it is difficult to understand which class will actually handle the property.

In the case of a private property, the different instances will stay distinct. In the case of protected or public properties, they will all share the same value.

It is recommended to avoid redefining the same property in a hierarchy.

Short name	Classes/RedefinedProperty
Rulesets	*ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.503 Reflection Export() Is Deprecated

export() method in Reflection classes is now deprecated. It is obsolete since PHP 7.4 and will disappear in PHP 8.0.

The Reflector interface, which is implemented by all reflection classes, specifies two methods: __toString() and export().

```
<?php

ReflectionFunction::export('foo');
```

(continues on next page)

(continued from previous page)

```
// same as
echo new ReflectionFunction('foo'), \n;

$str = ReflectionFunction::export('foo', true);
// same as
$str = (string) new ReflectionFunction('foo');

?>
```

See also Reflection export() methods and Reflection.

9.503.1 Suggestions

- Cast the object to string
- Remove the call to export()

Short name	Php/ReflectionExportIsDeprecated
Rulesets	*CompatibilityPHP74*
Severity	Minor
Time To Fix	Quick (30 mins)

9.504 Regex On Arrays

Avoid using a loop with arrays of regex or values. There are several PHP function which work directly on arrays, and much faster.

preg_grep() is able to extract all matching strings from an array, or non-matching strings. This usually saves a loop over the strings.

preg_filter() is able to extract all strings from an array, matching at least one regex in an array. This usually saves a double loop over the strings and the regex. The trick here is to provide '$0' as replacement, leading preg_filter() to replace the found string by itself.

Finally, preg_replace_callback() an preg_replace_callback_array() are also able to apply an array of regex to an array of strings, and then, apply callbacks to the found values.

```
<?php

$regexs = ['/ab+c/', '/abd+/', '/abe+/'];
$strings = ['/abbbbc/', '/abd/', '/abeee/'];

// Directly extract all strings that match one regex
foreach($regexs as $regex) {
    $results[] = preg_grep($regex, $strings);
}

// extract all matching regex, by string
foreach($strings as $string) {
    $results[] = preg_filter($regexs, array_fill(0, count($regexs), '$0'), $string);
}

// very slow way to get all the strings that match a regex
```

(continues on next page)

(continued from previous page)

```
foreach($regexs as $regex) {
    foreach($strings as $string) {
        if (preg_match($regex, $string)) {
            $results[] = $string;
        }
    }
}

?>
```

See also preg_filter.

9.504.1 Suggestions

- Apply preg_match() to an array of string or regex, via preg_filter() or preg_grep().
- Apply preg_match() to an array of string or regex, via preg_replace_callback() or preg_replace_callback_array().

Short name	Performances/RegexOnArrays
Rulesets	*Performances*
Severity	Minor
Time To Fix	Quick (30 mins)

9.505 Register Globals

`register_globals` was a PHP directive that dumped all incoming variables from GET, POST, COOKIE and FILES as global variables in the called scripts. This lead to security failures, as the variables were often used but not filtered.

Though it is less often found in more recent code, `register_globals` is sometimes needed in legacy code, that haven't made the move to eradicate this style of coding. Backward compatible pieces of code that mimic the `register_globals` features usually create even greater security risks by being run after scripts startup. At that point, some important variables are already set, and may be overwritten by the incoming call, creating confusion in the script.

Mimicking `register_globals` is achieved with variables variables, extract(), parse_str() and import_request_variables() (Up to PHP 5.4).

```
<?php

// Security warning ! This overwrites existing variables.
extract($_POST);

// Security warning ! This overwrites existing variables.
foreach($_REQUEST as $var => $value) {
    $$var = $value;
}

?>
```

9.505.1 Suggestions

- Avoid reimplementing register_globals
- Use a container to store and access commonly used values

Short name	Security/RegisterGlobals
Rulesets	*Security*
Severity	Critical
Time To Fix	Slow (1 hour)
Examples	*TeamPass*, *XOOPS*

9.506 Relay Function

Relay function only delegate workload to another one.

Relay functions and methods are delegating the actual work to another function or method. They do not have any impact on the results, besides exposing another name for the same feature.

```
<?php

function myStrtolower($string) {
    return \strtolower($string);
}

?>
```

Relay functions are typical of transition API, where an old API have to be preserved until it is fully migrated. Then, they may be removed, so as to reduce confusion, and simplify the API.

9.506.1 Suggestions

- Remove relay function, call directly the final function
- Remove the target function, and move the code here
- Add more logic to that function, like conditions or cache

Short name	Functions/RelayFunction
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*TeamPass*, *SPIP*

9.507 Repeated Interface

A class should implements only once an interface. An interface can only extends once another interface. In both cases, parent classes or interfaces must be checked.

PHP accepts multiple times the same interface in the `implements` clause. In fact, it doesn't do anything beyond the first implement.

```php
<?php

use i as j;

interface i {}

// Multiple ways to reference an interface
class foo implements i, \i, j {}

// This applies to interfaces too
interface bar extends i, \i, j {}

?>
```

This code may compile, but won't execute.

See also Object Interfaces and The Basics.

9.507.1 Suggestions

- Remove the interface usage at the lowest class or interface

Short name	Interfaces/RepeatedInterface
Rulesets	*Analyze*, *LintButWontExec*
Severity	Minor
Time To Fix	Instant (5 mins)

9.508 Repeated Regex

Repeated regex should be centralized.

When a regex is repeatedly used in the code, it is getting harder to update.

```php
<?php

// Regex used several times, at least twice.
preg_match('/^abc_|^square$/i', $_GET['x']);

//.......

preg_match('/^abc_|^square$/i', $row['name']);

// This regex is dynamically built, so it is not reported.
preg_match('/^circle|^'.$x.'$/i', $string);

// This regex is used once, so it is not reported.
preg_match('/^circle|^square$/i', $string);

?>
```

Regex that are repeated at least once (aka, used twice or more) are reported. Regex that are dynamically build are not reported.

9.508.1 Suggestions

- Create a central library of regex
- Use the regex inventory to spot other regex that are close, and should be identical.

Short name	Structures/RepeatedRegex
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Vanilla*, *Tikiwiki*

9.509 Repeated print()

Always merge several print or echo in one call.

It is recommended to use echo with multiple arguments, or a concatenation with print, instead of multiple calls to print echo, when outputting several blob of text.

```
<?php

//Write :
  echo 'a', $b, 'c';
  print 'a' . $b . 'c';

//Don't write :
  print 'a';
  print $b;
  print 'c';
?>
```

9.509.1 Suggestions

- Merge all prints into one echo call, separating arguments by commas.
- Collect all values in one variable, and do only one call to print or echo.

Short name	Structures/RepeatedPrint
Rulesets	*Analyze*, *Suggestions*, *Top10*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	no-repeated-print
Examples	*Edusoho*, *HuMo-Gen*

9.510 Reserved Keywords In PHP 7

PHP reserved names for class/trait/interface. They won't be available anymore in user space starting with PHP 7.

For example, string, float, false, true, null, resource,'... <https://www.php.net/manual/en/functions.arguments.php#functions.variable-arg-list>'_ are not acceptable as class name.

```
<?php

// This doesn't compile in PHP 7.0 and more recent
class null { }

?>
```

See also List of other reserved words.

9.510.1 Suggestions

- Avoid using PHP reserved keywords

Short name	Php/ReservedKeywords7
Rulesets	*CompatibilityPHP70*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.511 Results May Be Missing

preg_match() may return empty values, if the search fails. It is important to check for the existence of results before assigning them to another variable, or using it.

```
<?php
    preg_match('/PHP ([0-9\.]+) /', $res, $r);
    $s = $r[1];
    // $s may end up null if preg_match fails.
?>
```

Short name	Structures/ResultMayBeMissing
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)

9.512 Rethrown Exceptions

Throwing a caught exception is usually useless and dead code.

When exceptions are caught, they should be processed or transformed, but not rethrown as is.

Those issues often happen when a catch structure was positioned for debug purposes, but lost its usage later.

```
<?php

try {
    doSomething();
} catch (Exception $e) {
    throw $e;
```

(continues on next page)

(continued from previous page)

```
}

?>
```

See also What are the best practices for catching and re-throwing exceptions? and Exception chaining.

9.512.1 Suggestions

- Log the message of the exception for later usage.
- Remove the try/catch and let the rest of the application handle this exception.
- Chain the exception, by throwing a new exception, including the caught exception.

Short name	Exceptions/Rethrown
Rulesets	*Dead code*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*PrestaShop*

9.513 Return True False

These conditional expressions return true/false, depending on the condition. This may be simplified by dropping the control structure altogether.

```
<?php

if (version_compare($a, $b) >= 0) {
    return true;
} else {
    return false;
}

?>
```

This may be simplified with :

```
<?php

return version_compare($a, $b) >= 0;

?>
```

This may be applied to assignations and ternary operators too.

```
<?php

if (version_compare($a, $b) >= 0) {
    $a = true;
} else {
    $a = false;
}
```

(continues on next page)

(continued from previous page)

```
$a = version_compare($a, $b) >= 0 ? false : true;

?>
```

9.513.1 Suggestions

- Return directly the comparison, without using the if/then structure
- Cast the value to (boolean) and use it instead of the ternary

Short name	Structures/ReturnTrueFalse
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Mautic*, *FuelCMS*

9.514 Return With Parenthesis

return statement doesn't need parenthesis. PHP tolerates them with return statement, but it is recommended not to use them.

From the PHP Manual : 'Note: Note that since return is a language construct and not a function, the parentheses surrounding its argument are not required and their use is discouraged.'.

```
<?php

function foo() {
    $a = rand(0, 10);

    // No need for parenthesis
    return $a;

    // Parenthesis are useless here
    return ($a);

    // Parenthesis are useful here: they are needed by the multplication.
    return ($a + 1) * 3;
}

?>
```

See also PHP return(value); vs return value; and return.

9.514.1 Suggestions

- Remove the parenthesis

Short name	Php/ReturnWithParenthesis
Rulesets	*Coding Conventions*, *Suggestions*
Severity	Minor
Time To Fix	Instant (5 mins)

9.515 Reuse Variable

A variable is already holding the content that is calculated multiple times over.

It is recommended to use the cached value. This saves some computation, in particular when used in a loop, and speeds up the process.

```
<?php

function foo($a) {
    $b = strtolower($a);

    // strtolower($a) is already calculated in $b. Just reuse the value.
    if (strtolower($a) === 'c') {
        doSomething();
    }
}

?>
```

9.515.1 Suggestions

- Reuse the already created variable

Short name	Structures/ReuseVariable
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Medium

9.516 Safe Curl Options

It is advised to always use CURLOPT_SSL_VERIFYPEER and CURLOPT_SSL_VERIFYHOST when requesting a SSL connection.

With those tests, the certificate is verified, and if it isn't valid, the connection fails : this is a safe behavior.

```
<?php
$ch = curl_init();

curl_setopt($ch, CURLOPT_URL, https://www.php.net/);

// To be safe, always set this to true
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, true);

```

(continues on next page)

(continued from previous page)

```
curl_exec($ch);
curl_close($ch);
?>
```

See also Don’t turn off CURLOPT_SSL_VERIFYPEER, fix your PHP configuration, Certainty: Automated CACert.pem Management for PHP Software and Server-Side HTTPS Requests.

9.516.1 Suggestions

- Always use CURLOPT_SSL_VERIFYPEER and HTTPS for communication with other servers

Short name	Security/CurlOptions
Rulesets	*Security*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*OpenConf*

9.517 Safe HTTP Headers

Avoid configuring HTTP headers with lax restriction from within PHP.

There are a lot of HTTP headers those days, targeting various vulnerabilities. To ensure backward compatibility, those headers have a default mode that is lax and permissive. It is recommended to avoid using those from within the code.

```
<?php

//Good configuration, limiting access to origin
header('Access-Control-Allow-Origin: https://www.exakat.io');

//Configuration is present, but doesn't restrict anything : any external site is a␣
↪potential source
header('Access-Control-Allow-Origin: *');

?>
```

See also Hardening Your HTTP Security Headers, How To Secure Your Web App With HTTP Headers and Security-Headers.

9.517.1 Suggestions

- Remove usage of those headers

Short name	Security/SafeHttpHeaders
Rulesets	*Security*
Severity	Major
Time To Fix	Quick (30 mins)

9.518 Same Conditions In Condition

At least two consecutive if/then structures use identical conditions. The latter will probably be ignored.

This analysis returns false positive when there are attempt to fix a situation, or to call an alternative solution.

Conditions that are shared between if structures, but inside a logical OR expression are also detected.

```php
<?php

if ($a == 1) { doSomething(); }
elseif ($b == 1) { doSomething(); }
elseif ($c == 1) { doSomething(); }
elseif ($a == 1) { doSomething(); }
else {}

// Also works on if then else if chains
if ($a == 1) { doSomething(); }
else if ($b == 1) { doSomething(); }
else if ($c == 1) { doSomething(); }
else if ($a == 1) { doSomething(); }
else {}

// Also works on if then else if chains
// Here, $a is common and sufficient in both conditions
if ($a || $b) { doSomething(); }
elseif ($a || $c) { doSomethingElse(); }

// This sort of situation generate false postive.
$config = load_config_from_commandline();
if (empty($config)) {
    $config = load_config_from_file();
    if (empty($config)) {
        $config = load_default_config();
    }
}

?>
```

9.518.1 Suggestions

- Merge the two conditions into one
- Make the two conditions different

Short name	Structures/SameConditions
Rulesets	*Analyze*, *CI-checks*
Severity	Critical
Time To Fix	Quick (30 mins)
Examples	*TeamPass*, *Typo3*

9.519 Same Variable Foreach

A foreach which uses its own source as a blind variable is actually broken.

Actually, PHP makes a copy of the source before it starts the loop. As such, the same variable may be used for both source and blind value.

Of course, this is very confusing, to see the same variables used in very different ways.

The source will also be destroyed immediately after the blind variable has been turned into a reference.

```
<?php

$array = range(0, 10);
foreach($array as $array) {
    print $array.PHP_EOL;
}

print_r($array); // display number from 0 to 10.

$array = range(0, 10);
foreach($array as &$array) {
    print $array.PHP_EOL;
}

print_r($array); // display 10

?>
```

Short name	Structures/AutoUnsetForeach
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.520 Scalar Are Not Arrays

It is wrong to use a scalar as an array, a Warning is emitted. PHP 7.4 emits a Warning in such situations.

```
<?php

// Here, $x may be null, and in that case, the echo will fail.
function foo(?A $x) {
    echo $x[2];
}

?>
```

Typehinted argument with a scalar are reported by this analysis. Also, nullable arguments, both with typehint and return type hint.

See also E_WARNING for invalid container read array-access.

9.520.1 Suggestions

- Update type hints to avoid scalar values
- Remove the array syntax in the code using the results

Short name	Php/ScalarAreNotArrays
Rulesets	*Analyze*, *CompatibilityPHP74*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

9.521 Scalar Or Object Property

Property shouldn't use both object and scalar syntaxes. When a property may be an object, it is recommended to implement the Null Object pattern : instead of checking if the property is scalar, make it always object.

```php
<?php

class x {
    public $display = 'echo';

    function foo($string) {
        if (is_string($this->display)) {
            echo $this->string;
        } elseif ($this->display instanceof myDisplayInterface) {
            $display->display();
        } else {
            print Error when displaying\n;
        }
    }
}

interface myDisplayInterface {
    public function display($string); // does the display in its own way
}

class nullDisplay implements myDisplayInterface {
    // implements myDisplayInterface but does nothing
    public function display($string) {}
}

class x2 {
    public $display = null;

    public function __construct() {
        $this->display = new nullDisplay();
    }

    function foo($string) {
        // Keep the check, as $display is public, and may get wrong values
        if ($this->display instanceof myDisplayInterface) {
            $display->display();
        } else {
            print Error when displaying\n;
        }
    }
}

// Simple class for echo
```

(continues on next page)

(continued from previous page)

```
class echoDisplay implements myDisplayInterface {
    // implements myDisplayInterface but does nothing
    public function display($string) {
        echo $string;
    }
}

?>
```

See also Null Object Pattern. and The Null Object Pattern.

9.521.1 Suggestions

- Only use one type of syntax with your properties.

Short name	Classes/ScalarOrObjectProperty
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*SugarCrm*

9.522 Self Using Trait

Trait uses itself : this is unnecessary. Traits may use themselves, or be used by other traits, that are using the initial trait itself.

PHP handles the situation quietly, by ignoring all extra use of the same trait, keeping only one valid version.

```
<?php

// empty, but valid
trait a {}

// obvious self usage
trait b { use b; }

// less obvious self usage
trait c { use d, e, f, g, h, c; }

// level 2 self usage
trait i { use j; }
trait j { use i; }

?>
```

See also Traits.

9.522.1 Suggestions

- Remove the extra usage of the trait.

Short name	Traits/SelfUsingTrait
Rulesets	*Dead code*, *ClassReview*
Severity	Minor
Time To Fix	Slow (1 hour)

9.523 Semantic Typing

Arguments names are only useful inside the method's body. They are not actual type.

```
<?php

// arguments should be a string and an array
function foo($array, $str) {
    // more code
    return $boolean;
}

// typehint is actually checking the values
function bar(iterable $closure) : bool {
    // more code
    return true;
}

?>
```

9.523.1 Suggestions

- Use a typehint to make sure the argument is of the expected type.

Short name	Functions/SemanticTyping
Rulesets	*Semantics*
Severity	Minor
Time To Fix	Quick (30 mins)

9.524 Session Lazy Write

Classes that implements SessionHandlerInterface must also implements SessionUpdateTimestampHandlerInterface.

The two extra methods are used to help lazy loading : the first actually checks if a sessionId is available, and the seconds updates the time of last usage of the session data in the session storage.

This was spotted by `Nicolas Grekas`, and fixed in Symfony [HttpFoundation] Make sessions secure and lazy #24523.

```
<?php

interface SessionUpdateTimestampHandlerInterface {
    // returns a boolean to indicate that valid data is available for this sessionId,␣
↪or not.
    function validateId($sessionId);
```

(continues on next page)

(continued from previous page)

```
    //called to change the last time of usage for the session data.
    //It may be a file's touch or full write, or a simple update on the database
    function updateTimestamp($sessionId, $sessionData);
}

?>
```

See also Sessions: Improve original RFC about lazy_write and the Sessions.

9.524.1 Suggestions

- Implements the SessionUpdateTimestampHandlerInterface interface

Short name	Security/SessionLazyWrite
Rulesets	*Security*
Severity	Major
Time To Fix	Slow (1 hour)

9.525 Set Aside Code

Setting aside code should be made into a method.

Setting aside code happens when one variable or member is stored locally, to be temporarily replaced by another value. Once the new value has been processed, the original value is reverted.

The temporary change of the value makes the code hard to read.

It is a good example of a piece of code that could be moved to a separate method or function. Using the temporary value as a parameter makes the change visible, and avoid local pollution.

```
<?php

// Setting aside database
class cache extends Storage {
    private $database = null;

    function __construct($database) {
        $this->database = $database;
    }

    function foo($values) {
        // handling storage with sqlite3
        $secondary = new cache(new Sqlite3(':memory:'));
        $secondary->store($values);

        $this->store($values);      // handling storage with injection
    }
}

// Setting aside database to cache data in two distinct backend
class cache extends Storage {
    private $database = null;
```

(continues on next page)

(continued from previous page)

```
    function __construct(\Pdo $database) {
        $this->database = $database;
    }

    function foo($values) {
        // $this->database is set aside for secondary configuration
        $side = $this->database;
        $this->database = new Sqlite3(':memory:');
        $this->store($values);      // handling storage with sqlite3
        $this->database = $side;
        // $this->database is restored
        $this->store($values);      // handling storage with injection
    }
}

?>
```

9.525.1 Suggestions

- Extract the code that run with the temporary value to a separate method.

Short name	Structures/SetAside
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.526 Set Cookie Safe Arguments

The last five arguments of setcookie() and setrawcookie() are for security. Use them anytime you can.

`setcookie ( string $name [, string $value = [, int $expire = 0 [, string $path = [, string $domain = [, bool $secure = false [, bool $httponly = false ]]]]]] )`

The `$expire` argument sets the date of expiration of the cookie. It is recommended to make it as low as possible, to reduce its chances to be captured. Sometimes, low expiration date may be several days (for preferences), and other times, low expiration date means a few minutes.

The `$path` argument limits the transmission of the cookie to URL whose path matches the one mentioned here. By default, it is `'/'`, which means the whole server. If a cookie usage is limited to a part of the application, use it here.

The `$domain` argument limits the transmission of the cookie to URL whose domain matches the one mentioned here. By default, it is `''`, which means any server on the internet. At worse, you may use `mydomain.com` to cover your whole domain, or better, refine it with the actual subdomain of usage.

The `$secure` argument limits the transmission of the cookie over HTTP (by default) or HTTPS. The second is better, as the transmission of the cookie is crypted. In case HTTPS is still at the planned stage, use '$_SERVER[HTTPS]'. This environment variable is false on HTTP, and true on HTTPS.

The `$httponly` argument limits the access of the cookie to JavaScript. It is only transmitted to the browser, and retransmitted. This helps reducing XSS and CSRF attacks, though it is disputed.

The $samesite argument limits the sending of the cookie to the domain that initiated the request. It is by default Lax but should be upgraded to Strict whenever possible. This feature is available as PHP 7.3.

```
<?php

//admin cookie, available only on https://admin.my-domain.com/system/, for the next␣
↪minute, and not readable by javascript
setcookie(admin, $login, time()+60, /system/, admin.my-domain.com, $_SERVER['HTTPS'],␣
↪1);

//login cookie, available until the browser is closed, over http or https
setcookie(login, $login);

//removing the login cookie : Those situations are omitted by the analysis
setcookie(login, '');

?>
```

See also setcookie and 'SameSite' cookie attribute.

9.526.1 Suggestions

- Use all the argument when setting cookies with PHP functions

Short name	Security/SetCookieArgs
Rulesets	*Security*
Severity	Major
Time To Fix	Instant (5 mins)

9.527 Setlocale() Uses Constants

setlocal() don't use strings but constants.

The first argument of setlocale() must be one of the valid constants, LC_ALL, LC_COLLATE, LC_CTYPE, LC_MONETARY, LC_NUMERIC, LC_TIME, LC_MESSAGES.

```
<?php

// Use constantes for setlocale first argument
setlocale(LC_ALL, 'nl_NL');
setlocale(\LC_ALL, 'nl_NL');

// Don't use string for setlocale first argument
setlocale('LC_ALL', 'nl_NL');
setlocale('LC_'.'ALL', 'nl_NL');

?>
```

The PHP 5 usage of strings (same name as above, enclosed in ' or ") is not legit anymore in PHP 7 and later.

See also setlocale.

Short name	Structures/SetlocaleNeedsConstants
Rulesets	*CompatibilityPHP70*
Severity	Major
Time To Fix	Instant (5 mins)

9.528 Several Instructions On The Same Line

Usually, instructions do not share their line : one instruction, one line.

This is good for readability, and help at understanding the code. This is especially important when fast-reading the code to find some special situation, where such double-meaning line way have an impact.

```php
<?php

switch ($x) {
    // Is it a fallthrough or not ?
    case 1:
        doSomething(); break;

    // Easily spotted break.
    case 1:
        doSomethingElse();
        break;

    default :
        doDefault();
        break;
}

?>
```

See also Object Calisthenics, rule # 5.

Short name	Structures/OneLineTwoInstructions
Rulesets	*Analyze*
Severity	Major
Time To Fix	Instant (5 mins)
Examples	*Piwigo*, *Tine20*

9.529 Short Open Tags

Usage of short open tags is discouraged. The following files were found to be impacted by the short open tag directive at compilation time. They must be reviewed to ensure no &lt;? tags are found in the code.

Short name	Php/ShortOpenTagRequired
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)

9.530 Short Syntax For Arrays

Arrays written with the new short syntax.

PHP 5.4 introduced the new short syntax, with square brackets. The previous syntax, based on the array() keyword is still available.

```
<?php

// All PHP versions array
$a = array(1, 2, 3);

// PHP 5.4+ arrays
$a = [1, 2, 3];

?>
```

See also Array.

Short name	Arrays/ArrayNSUsage
Rulesets	*CompatibilityPHP53*
Severity	Critical
Time To Fix	Quick (30 mins)

9.531 Should Be Single Quote

Use single quote for simple strings.

Static content inside a string, that has no single quotes nor escape sequence (such as n or t), should be using single quote delimiter, instead of double quote.

```
<?php

$a = abc;

// This one is using a special sequence
$b = cde\n;

// This one is using two special sequences
$b = \x03\u{1F418};

?>
```

If you have too many of them, don't loose your time switching them all. If you have a few of them, it may be good for consistence.

Short name	Type/ShouldBeSingleQuote
Rulesets	*Coding Conventions*
Severity	Minor
Time To Fix	Instant (5 mins)
ClearPHP	no-double-quote

9.532 Should Chain Exception

Chain exception to provide more context.

When catching an exception and rethrowing another one, it is recommended to chain the exception : this means providing the original exception, so that the final recipient has a chance to track the origin of the problem. This doesn't change the thrown message, but provides more information.

Note : Chaining requires PHP > 5.3.0.

```
<?php
    try {
        throw new Exception('Exception 1', 1);
    } catch (\Exception $e) {
        throw new Exception('Exception 2', 2, $e);
        // Chaining here.

    }
?>
```

See also Exception::'__construct <https://www.php.net/manual/en/exception.construct.php>'_ and What are the best practices for catching and re-throwing exceptions?.

9.532.1 Suggestions

- Add the incoming exception to the newly thrown exception

Short name	Structures/ShouldChainException
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Magento*, *Tine20*

9.533 Should Deep Clone

By default, PHP makes a shallow clone. It only clone the scalars, and keep the reference to any object already referenced. This means that the cloned object and its original share any object they hold as property.

This is where the magic method __clone() comes into play. It is called, when defined, at clone time, so that the cloned object may clone all the needed sub-objects.

It is recommended to use the __clone() method whenever the objects hold objects.

```
<?php

class a {
    public $b = null;

    function __construct() {
        $this->b =  new Stdclass();
        $this->b->c = 1;
    }
}
```

(continues on next page)

(continued from previous page)

```
class ab extends a {
    function __clone() {
        $this->b = clone $this->b;
    }
}

// class A is shallow clone, so $a->b is not cloned
$a = new a();
$b = clone $a;
$a->b->c = 3;
echo $b->b->c;
// displays 3

// class Ab is deep clone, so $a->b is cloned
$a = new ab();
$b = clone $a;
$a->b->c = 3;
echo $b->b->c;
// displays 1

?>
```

See also PHP Clone and Shallow vs Deep Copying and Cloning objects.

9.533.1 Suggestions

-

Short name	Classes/ShouldDeepClone
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.534 Should Have Destructor

PHP destructors are called when the object has to be destroyed. By default, PHP calls recursively the destructor on internal objects, until everything is unset.

Unsetting objects and resources explicitly in the destructor is a good practice to reduce the amount of memory in use. It helps PHP resource counter to keep the numbers low, and easier to clean. This is a major advantage for long running scripts.

```
<?php

class x {
    function __construct() {
        $this->p = new y();
    }

    function __destruct() {
        print __METHOD__.PHP_EOL;
```

(continues on next page)

(continued from previous page)

```
        unset($this->p);
    }
}

class y {
    function __construct() {
        print __METHOD__.PHP_EOL;
        $this->p = new y();
    }

    function __destruct() {
        print __METHOD__.PHP_EOL;
        unset($this->p);
    }
}

$a = (new x);
sleep(1);

// This increment the resource counter by one for the property.
$p = $a->p;
unset($a);
sleep(3);

print 'end'.PHP_EOL;
// Y destructor is only called here, as the object still exists in $p.

?>
```

See also Destructor, and Php Destructors.

9.534.1 Suggestions

- Add a destruct method to the class to help clean at destruction time.

Short name	Classes/ShouldHaveDestructor
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Slow (1 hour)

9.535 Should Make Alias

Long names should be aliased.

Aliased names are easy to read at the beginning of the script; they may be changed at one point, and update the whole code at the same time. Finally, short names makes the rest of the code readable.

```
<?php

namespace x\y\z;

use a\b\c\d\e\f\g as Object;
```

(continues on next page)

(continued from previous page)

```php
// long name, difficult to read, prone to change.
new a\b\c\d\e\f\g();

// long name, difficult to read, prone to silent dead code if namespace change.
if ($o instanceof a\b\c\d\e\f\g) {

}

// short names Easy to update all at once.
new Object();
if ($o instanceof Object) {

}

?>
```

Short name	Namespaces/ShouldMakeAlias
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.536 Should Make Ternary

Ternary operators are the best when assigning values to a variable.

This way, they are less verbose, compatible with assignation and easier to read.

```php
<?php
    // verbose if then structure
    if ($a == 3) {
        $b = 2;
    } else {
        $b = 3;
    }

    // compact ternary call
    $b = ($a == 3) ? 2 : 3;

    // verbose if then structure
    // Works with short assignations and simple expressions
    if ($a != 3) {
        $b += 2 - $a * 4;
    } else {
        $b += 3;
    }

    // compact ternary call
    $b += ($a != 3) ? 2 - $a * 4 : 3;

?>
```

Short name	Structures/ShouldMakeTernary
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)

9.537 Should Preprocess Chr()

Replace literal chr() calls with their escape sequence.

chr() is a functioncall, that cannot be cached. It is only resolved at execution time. On the other hand, literal values are preprocessed by PHP and may be cached.

```
<?php

// This is easier on PHP
$a = "0000 is great!";

// This is slow
$a = chr(80), chr(72), chr(80), chr(32), ' is great!';

// This would be the best with this example, but it is not always possible
$a = 'PHP is great!';

?>
```

This is a micro-optimisation.

See also Escape sequences.

9.537.1 Suggestions

- Use PHP string sequences, and skip chr() at execution time

Short name	Php/ShouldPreprocess
Rulesets	none
Examples	*phpadsnew*

9.538 Should Typecast

When typecasting, it is better to use the casting operator, such as `(int)` or `(bool)`.

Functions such as intval() or settype() are always slower.

```
<?php

// Fast version
$int = (int) $X;

// Slow version
$int = intval($X);

```

(continues on next page)

(continued from previous page)

```
// Convert to base 8 : can't use (int) for that
$int = intval($X, 8);


?>
```

This is a micro-optimisation, although such conversion may be use multiple time, leading to a larger performance increase.

Note that intval() may also be used to convert an integer to another base.

9.538.1 Suggestions

- Use a typecast, instead of a functioncall.

Short name	Type/ShouldTypecast
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*xataface*, *OpenConf*

9.539 Should Use Coalesce

PHP 7 introduced the `??` operator, that replaces longer structures to set default values when a variable is not set.

```
<?php

// Fetches the request parameter user and results in 'nobody' if it doesn't exist
$username = $_GET['user'] ?? 'nobody';
// equivalent to: $username = isset($_GET['user']) ? $_GET['user'] : 'nobody';

// Calls a hypothetical model-getting function, and uses the provided default if it␣
↪fails
$model = Model::get($id) ?? $default_model;
// equivalent to: if (($model = Model::get($id)) === NULL) { $model = $default_model;␣
↪}

?>
```

Sample extracted from PHP docs Isset Ternary.

See also New in PHP 7: null coalesce operator.

9.539.1 Suggestions

- Replace the long syntax with the short one

Short name	Php/ShouldUseCoalesce
Rulesets	*Analyze*, *Suggestions*, *CI-checks*
Php Version	With PHP 7.0 and more recent
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*ChurchCRM*, *Cleverstyle*

9.540 Should Use Constants

The following functions have related constants that should be used as arguments, instead of scalar literals, such as integers or strings.

```
<?php

// The file is read and new lines are ignored.
$lines = file('file.txt', FILE_IGNORE_NEW_LINES)

// What is this doing, with 2 ?
$lines = file('file.txt', 2);

?>
```

See also Bitmask Constant Arguments in PHP.

9.540.1 Suggestions

- Use PHP native constants whenever possible, for better readability.

Short name	Functions/ShouldUseConstants
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Tine20*

9.541 Should Use Explode Args

explode() has a third argument, which limits the amount of exploded elements. With it, it is possible to collect only the first elements, or drop the last ones.

```
<?php

$exploded = explode(DELIMITER, $string);

// use explode(DELIMITER, $string, -1);
array_pop($exploded);

// use explode(DELIMITER, $string, -2);
$c = array_slice($exploded, 0, -2);
```

(continues on next page)

(continued from previous page)

```
// with explode()'s third argument :
list($a, $b) = explode(DELIMITER, $string, 2);

// with list() omitted arguments
list($a, $b, ) = explode(DELIMITER, $string);

?>
```

See also explode.

9.541.1 Suggestions

-

Short name	Structures/ShouldUseExplodeArgs
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.542 Should Use Foreach

Use foreach instead of for when traversing an array.

Foreach() is the modern loop : it maps automatically every element of the array to a blind variable, and loop over it. This is faster and safer.

```
<?php

// Foreach version
foreach($array as $element) {
    doSomething($element);
}

// The above case may even be upgraded with array_map and a callback,
// for the simplest one of them
$array = array_map('doSomething', $array);

// For version (one of various alternatives)
for($i = 0; $i < count($array); $i++) {
    $element = $array[$i];
    doSomething($element);
}

// Based on array_pop or array_shift()
while($value = array_pop($array)) {
    doSomething($array);
}

?>
```

See also foreach and 5 Ways To Loop Through An Array In PHP.

9.542.1 Suggestions

- Move for() loops to foreach(), whenever they apply to a finite list of elements

Short name	Structures/ShouldUseForeach
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*ExpressionEngine*, *Woocommerce*

9.543 Should Use Function

Functioncalls that fall back to global scope should be using 'use function' or be fully namespaced.

PHP searches for functions in the local namespaces, and in case it fails, makes the same search in the global scope. Anytime a native function is referenced this way, the search (and fail) happens. This slows down the scripts.

The speed bump range from 2 to 8 %, depending on the availability of functions in the local scope. The overall bump is about 1 µs per functioncall, which makes it a micro optimisation until a lot of function calls are made.

Based on one of Marco Pivetta tweet.

```
<?php

namespace X {
    use function strtolower as strtolower_aliased;

    // PHP searches for strtolower in X, fails, then falls back to global scope,␣
↪succeeds.
    $a = strtolower($b);

    // PHP searches for strtolower in global scope, succeeds.
    $a = \strtolower($b);

    // PHP searches for strtolower_aliased in global scope, succeeds.
    $a = \strtolower_aliased($b);
}

?>
```

This analysis is a related to Performances/Php74ArrayKeyExists, and is a more general version.

See also blog post.

9.543.1 Suggestions

- Use the *use* command for arrray_key_exists(), at the beginning of the script
- Use an initial before array_key_exists()
- Remove the namespace

Short name	Php/ShouldUseFunction
Rulesets	*Performances*
Severity	Minor
Time To Fix	Slow (1 hour)

9.544 Should Use Local Class

Methods should use the defining class, or be functions.

Methods should use `$this` with another method or a property, or call `parent\:\:`. Static methods should call another static method, or a static property. Methods which are overwritten by a child class are omitted : the parent class act as a default value for the children class, and this is correct.

```
<?php

class foo {
    public function __construct() {
        // This method should do something locally, or be removed.
    }
}

class bar extends foo {
    private $a = 1;

    public function __construct() {
        // Calling parent:: is sufficient
        parent::__construct();
    }

    public function barbar() {
        // This is acting on the local object
        $this->a++;
    }

    public function barfoo($b) {
        // This has no action on the local object. It could be a function or a␣
↪closure where needed
        return 3 + $b;
    }
}

?>
```

Note that a method using a class constant is not considered as using the local class, for this analyzer.

9.544.1 Suggestions

- Make this method a function
- Actually use $this, or any related attributes of the class

Short name	Classes/ShouldUseThis
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
ClearPHP	not-a-method

9.545 Should Use Math

Use math operators to make the operation readable.

```
<?php

// Adding one to self
$a *= 2;
// same as above
$a += $a;

// Squaring oneself
$a \*\*\= 2;
// same as above
$a *= $a;

// Removing oneself
$a = 0;
// same as above
$a -= $a;

// Dividing oneself
$a = 1;
// same as above
$a /= $a;

// Divisition remainer
$a = 0;
// same as above
$a %= $a;

?>
```

See also Mathematical Functions.

9.545.1 Suggestions

- Use explicit math assignation

Short name	Structures/ShouldUseMath
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*OpenEMR*

9.546 Should Use Operator

Some functions duplicate the feature of an operator. When in doubt, it is better to use the operator.

Beware, some edge cases may apply. In particular, backward compatibility may prevent usage of newer features.

- array_push() is equivalent to []
- is_object() is equivalent to instanceof
- function_get_arg() and function_get_args() is equivalent to ellipsis : `...`
- chr() is equivalent to string escape sequences, such as `\n`, `\x69`, `u{04699}`
- call_user_func() is equivalent to `$functionName(arguments)`, `$object->$method(`. .. <https://www.php.net/manual/en/functions.arguments.php#functions.variable-arg-list>`_$arguments)`
- is_null() is equivalent to `=== null`
- php_version() is equivalent to `PHP_VERSION` (the constant)
- is_array(), is_int(), is_object(), etc. is equivalent to a scalar typehint

9.546.1 Suggestions

- Use [] instead of array_push()
- Use instanceof instead of is_object()
- Use ... instead of function_get_arg() and function_get_args()
- Use escape sequences instead of chr()
- Use dynamic function call instead of call_user_func()
- Use === null instead of is_null()
- Use PHP_VERSION instead of php_version()
- Use typehint instead of is_int(), is_string(), is_bool(), etc.

Short name	Structures/ShouldUseOperator
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Zencart*, *SugarCrm*

9.547 Should Use Prepared Statement

Modern databases provides support for prepared statement : it separates the query from the processed data and raise significantly the security.

Building queries with concatenations is not recommended, though not always avoidable. When possible, use prepared statements.

```
<?php
/* Execute a prepared statement by passing an array of values */

$sql = 'SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour';
$sth = $conn->prepare($sql, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
$sth->execute(array(':calories' => 150, ':colour' => 'red'));
$red = $sth->fetchAll();
?>
```

Same code, without preparation :

```
<?php

    $sql = 'SELECT name, color, calories FROM fruit WHERE calories < '.$conn-
↪quote(150).' AND colour = '.$conn->quotes('red').' ORDER BY name';
    $sth = $conn->query($sql) as $row);
}
?>
```

See also Prepared Statements, PHP MySQLi Prepared Statements Tutorial to Prevent SQL Injection, The Best Way to Perform MYSQLI Prepared Statements in PHP.

9.547.1 Suggestions

- Use an ORM
- Use an Active Record library
- Change the query to hard code it and make it not injectable

Name	Default	Type	Description
queryMethod	query_methods.json	data	Methods that call a query.

Short name	Security/ShouldUsePreparedStatement
Rulesets	*Analyze*, *Security*, *CI-checks*
Severity	Major
Time To Fix	Slow (1 hour)
Examples	*Dolibarr*

9.548 Should Use SetCookie()

Use setcookie() or setrawcookie(). Avoid using header() to do so, as the PHP native functions are more convenient and easier to spot during a refactoring.

setcookie() applies some encoding internally, for the value of the cookie and the date of expiration. Rarely, this encoding has to be skipped : then, use setrawencoding().

Both functions help by giving a checklist of important attributes to be used with the cookie.

```
<?php

// same as below
setcookie(myCookie, 'chocolate', time()+3600, /, , true, true);

// same as above. Slots for path and domain are omitted, but should be used whenever␣
↪possible
header('Set-Cookie: myCookie=chocolate; Expires='.date('r', (time()+3600)).'; Secure;␣
↪HttpOnly');

?>
```

See also Set-Cookie, setcookie.

9.548.1 Suggestions

- Use setcookie() function, instead of header()
- Use setcookie() function, instead of header()

Short name	Php/UseSetCookie
Rulesets	*Analyze*
Severity	Major
Time To Fix	Instant (5 mins)

9.549 Should Use array_column()

Avoid writing a whole slow loop, and use the native array_column().

array_column() is a native PHP function, that extract a property or a index from a array of object, or a multidimensional array. This prevents the usage of foreach to collect those values.

```
<?php

$a = array(array('b' => 1),
           array('b' => 2, 'c' => 3),
           array(          'c' => 4)); // b doesn't always exists

$bColumn = array_column($a, 'b');

// Slow and cumbersome code
$bColumn = array();
foreach($a as $k => $v) {
    if (isset($v['b'])) {
        $bColumn[] = $v['b'];
    }
}

?>
```

array_column() is faster than foreach() (with or without the isset() test) with 3 elements or more, and it is significantly faster beyond 5 elements. Memory consumption is the same.

See also [blog] ‘array_column() <https://benramsey.com/projects/array-column/>‘_.

9.549.1 Suggestions

- Use array_column(), instead of a foreach()

Short name	Php/ShouldUseArrayColumn
Rulesets	*Performances*, *Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.550 Should Use array_filter()

Should use array_filter().

array_filter() is a native PHP function, that extract elements from an array, based on a closure or a function. Using array_filter() shortens your code, and allows for reusing the filtering logic across the application, instead of hard coding it every time.

```
<?php

$a = range(0, 10); // integers from 0 to 10

// Extracts odd numbers
$odds = array_filter($a, function($x) { return $x % 2; });
$odds = array_filter($a, 'odd');

// Slow and cumbersome code for extracting odd numbers
$odds = array();
foreach($a as $v) {
    if ($a % 2) { // same filter than the closure above, or the odd function below
        $bColumn[] = $v;
    }
}

function foo($x) {
    return $x % 2;
}

?>
```

array_filter() is faster than foreach() (with or without the isset() test) with 3 elements or more, and it is significantly faster beyond 5 elements. Memory consumption is the same.

See also array_filter.

9.550.1 Suggestions

- Use array_filter()

Short name	Php/ShouldUseArrayFilter
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*xataface*, *shopware*

9.551 Should Use session_regenerateid()

session_regenerateid() should be used when sessions are used.

When using sessions, a session ID is assigned to the user. It is a random number, used to connect the user and its data on the server. Actually, anyone with the session ID may have access to the data. This is why those session ID are so long and complex.

A good approach to protect the session ID is to reduce its lifespan : the shorter the time of use, the better. While changing the session ID at every hit on the page may no be possible, a more reasonable approach is to change the session id when an important action is about to take place. What important means is left to the application to decide.

Based on this philosophy, a code source that uses ZendSession but never uses ZendSession::regenerateId() has to be updated.

```
<?php

    session_start();

    $id = (int) $_SESSION['id'];
    // no usage of session_regenerateid() anywhere triggers the analysis

    // basic regeneration every 20 hits on the page.
    if (++$_SESSION['count'] > 20) {
        session_regenerateid();
    }

?>
```

See session_regenerateid() and PHP Security Guide: Sessions.

9.551.1 Suggestions

- Add session_regenerateid() call before any important operation on the application

Short name	Security/ShouldUseSessionRegenerateId
Rulesets	*Security*
Severity	Major
Time To Fix	Slow (1 hour)

9.552 Should Yield With Key

iterator_to_array() will overwrite generated values with the same key.

PHP generators are based on the `yield` keyword. They also delegate some generating to other methods, with `yield from`.

When delegating, `yield from` uses the keys that are generated with `yield`, and otherwise, it uses auto-generated index, starting with 0.

The trap is that each `yield from` reset the index generation and start again with 0. Coupled with iterator_to_array(), this means that the final generated array may lack some values, while a foreach() loop would yield all of them.

```php
<?php

function g1() : Generator {
    for ( $i = 0; $i < 4; $i++ ) { yield $i; }
}

function g2() : Generator {
    for ( $i = 5; $i < 10; $i++ ) { yield $i; }
}

function aggregator() : Generator {
    yield from g1();
    yield from g2();
}

print_r(iterator_to_array());

/*
Array
(
    [0] => 6
    [1] => 7
    [2] => 8
    [3] => 9
    [4] => 4  // Note that 4 and 5 still appears
    [5] => 5  // They are not overwritten by the second yield
)
*/


foreach ( aggregator() as $i ) {
    print $i.PHP_EOL;
}

/*
0  // Foreach has no overlap and yield it all.
1
2
3
4
5
6
7
8
9
*/

?>
```

Thanks to Holger Woltersdorf for pointing this.

See also Generator syntax and Yielding values with keys.

9.552.1 Suggestions

- Use iterator_to_array() on each generator separately, and use array_merge() to merge all the arrays.
- Always yield with distinct keys

- Avoid iterator_to_array() and use foreach()

Short name	Functions/ShouldYieldWithKey
Rulesets	*Analyze*, *Top10*, *CI-checks*
Severity	Major
Time To Fix	Slow (1 hour)

9.553 Signature Trailing Comma

Trailing comma in method signature. This feature was added in PHP 8.0.

Allowing the trailing comma makes it possible to reduce the size of VCS's diff, when adding , removing a parameter.

```
<?php

// Example from the RFC
class Uri {
    private function __construct(
        ?string $scheme,
        ?string $user,
        ?string $pass,
        ?string $host,
        ?int $port,
        string $path,
        ?string $query,
        ?string $fragment // <-- ARGH!
    ) {
        ...
    }
}
?>
```

See also PHP RFC: Allow trailing comma in parameter list.

9.553.1 Suggestions

-

Short name	Php/SignatureTrailingComma
Rulesets	*CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*, *CompatibilityPHP74*
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)

9.554 Silently Cast Integer

Those are integer literals that are cast to a float when running PHP. They are too big for the current PHP version, and PHP resorts to cast them into a float, which has a much larger capacity but a lower precision.

Compare your literals to `PHP_MAX_INT` (typically `9223372036854775807`) and `PHP_MIN_INT` (typically `-9223372036854775808`). This applies to binary (`0b10101...`), octal (`0123123...`) and hexadecimal (`0xfffff...`) too.

```
<?php

echo 0b1010101101010110101011010101011010101011010101011010101011010111;
//6173123008118052203
echo 0b10101011010101101010110101010110101010110101010110101010110101111;
//1.2346246016236E+19

echo 0123123123123123123123;
//1498121094048818771
echo 01231231231231231231231;
//1.1984968752391E+19

echo 0x12309812311230;
//5119979279159856
echo 0x12309812311230fed;
//2.0971435127439E+19

echo 9223372036854775807; //PHP_MAX_INT
//9223372036854775807
echo 9223372036854775808;
9.2233720368548E+18

?>
```

See also Integer overflow.

9.554.1 Suggestions

- Make sure hexadecimal numbers have the right number of digits : generally, it is 15, but it may depends on your PHP version.

Short name	Type/SilentlyCastInteger
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*MediaWiki*

9.555 Similar Integers

This analysis reports all integer values that are expressed in different format.

```
<?php

// Three ways to write 10 (more available)
$a = 10;
$b = 012;
$x = 0xA;

// 7 is expressed in one way only
```

(continues on next page)

(continued from previous page)

```
$d = 7;
$d = 7;

// Four ways to write 11 (more available)
$a = 11;
$b = 013;
$x = 0xB;
$x = -+-11;

// Expressions are not counted

?>
```

9.555.1 Suggestions

-

Short name	Type/SimilarIntegers
Rulesets	*Coding Conventions*, *Semantics*
Severity	Minor
Time To Fix	Quick (30 mins)

9.556 Simple Global Variable

The global keyword should only be used with simple variables. Since PHP 7, it cannot be used with complex or dynamic structures.

```
<?php

// Forbidden in PHP 7
global $normalGlobal;

// Forbidden in PHP 7
global $$variable->global ;

// Tolerated in PHP 7
global ${$variable->global};

?>
```

Short name	Php/GlobalWithoutSimpleVariable
Rulesets	*CompatibilityPHP70*
Php Version	With PHP 7.0 and older
Severity	Critical
Time To Fix	Slow (1 hour)

9.557 Simple Switch

Switches are faster when relying only on integers or strings.

Since PHP 7.2, simple switches that use only strings or integers are optimized. The gain is as great as the switch is big.

```
<?php

// Optimized switch.
switch($b) {
    case "a":
        break;
    case "b":
        break;
    case "c":
        break;
    case "d":
        break;
    default :
        break;
}

// Unoptimized switch.
// Try moving the foo() call in the default, to keep the rest of the switch optimized.
switch($c) {
    case "a":
        break;
    case foo($b):
        break;
    case "c":
        break;
    case "d":
        break;
    default :
        break;
}

?>
```

See also PHP 7.2's "switch" optimisations.

9.557.1 Suggestions

- Split the switch between literal and dynamic cases
- Remove the dynamic cases from the switch

Short name	Performances/SimpleSwitch
Rulesets	*Performances*
Php Version	With PHP 7.2 and more recent
Severity	Major
Time To Fix	Quick (30 mins)

9.558 Simplify Regex

Avoid using regex when the searched string or the replacement are simple enough.

PRCE regex are a powerful way to search inside strings, but they also come at the price of performance. When the query is simple enough, try using strpos() or stripos() instead.

```
<?php

// simple preg calls
if (preg_match('/a/', $string))  {}
if (preg_match('/b/i', $string)) {} // case insensitive

// light replacements
if( strpos('a', $string)) {}
if( stripos('b', $string)) {}       // case insensitive

?>
```

9.558.1 Suggestions

- Use str_replace(), strtr() or even strpos()

Short name	Structures/SimplePreg
Rulesets	*Performances*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Zurmo*, *OpenConf*

9.559 Slice Arrays First

Always start by reducing an array before applying some transformation on it. The shorter array will be processed faster.

```
<?php

// fast version
$a = array_map('foo', array_slice($array, 2, 5));

// slower version
$a = array_slice(array_map('foo', $array), 2, 5);
?>
```

The gain produced here is greater with longer arrays, or greater reductions. They may also be used in loops. This is a micro-optimisation when used on short arrays.

9.559.1 Suggestions

- Use the array transforming function on the result of the array shortening function.

Short name	Arrays/SliceFirst
Rulesets	*Performances*, *Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*WordPress*

9.560 Slow Functions

Avoid using those slow native PHP functions, and replace them with alternatives.

```
<?php

$array = source();

// Slow extraction of distinct values
$array = array_unique($array);

// Much faster extraction of distinct values
$array = array_keys(array_count_values($array));

?>
```

Slow Function	Faster
array_diff() array_intersect() array_key_exists() array_map() array_search() array_udiff() array_uintersect() array_unshift() array_walk() in_array() preg_replace() strstr() uasort() uksort() usort() array_unique()	foreach() foreach() isset() and array_key_exists() foreach() array_flip() and isset() Use another way Use another way Use another way foreach() isset() strpos() strpos() Use another way Use another way Use another way array_keys() and array_count_values()

array_unique() has been accelerated in PHP 7.2 and may be used directly from this version on : Optimize ‘array_unique() <https://github.com/php/php-src/commit/6c2c7a023da4223e41fea0225c51a417fc8eb10d>‘_.

array_key_exists() has been accelerated in PHP 7.4 and may be used directly from this version on : Implement ZEND_ARRAY_KEY_EXISTS opcode to speed up ‘array_key_exists() <https://github.com/php/php-src/pull/3360>‘_.

9.560.1 Suggestions

- Replace the slow function with a faster version
- Remove the usage of the slow function

Short name	Performances/SlowFunctions
Rulesets	*Performances*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	avoid-those-slow-functions
Examples	*ChurchCRM*, *SuiteCrm*

9.561 Sqlite3 Requires Single Quotes

The escapeString() method from `SQLite3` doesn’t escape `"`, but only `'`.

```
<?php

// OK. escapeString is OK with '
$query = "SELECT * FROM table WHERE col = '".$sqlite->escapeString($x)."'";
```

(continues on next page)

(continued from previous page)

```

// This is vulnerable to " in $x
$query = 'SELECT * FROM table WHERE col = "'.$sqlite->escapeString($x).'"';

?>
```

To properly handle quotes and `NUL` characters, use bindParam() instead.

Quote from the PHP manual comments : `The reason this function doesn't escape double quotes is because double quotes are used with names (the equivalent of backticks in MySQL), as in table or column names, while single quotes are used for values.`

See also SQLite3::escapeString.

9.561.1 Suggestions

- Use prepared statements whenever possible
- Switch the query to use single quote

Short name	Security/Sqlite3RequiresSingleQuotes
Rulesets	*Security*
Severity	Major
Time To Fix	Slow (1 hour)

9.562 Static Global Variables Confusion

PHP can’t have variable that are both static and variable. While the syntax is legit, the variables will be alternatively global or static.

It is recommended to avoid using the same name for a global variable and a static variable.

```
<?php

function foo() {
    $a = 1; // $a is a local variable

    global $a; // $a is now a global variable

    static $a; // $a is not w static variable
}

?>
```

9.562.1 Suggestions

- Avoid using static variables
- Avoid using global variables
- Avoid using the same name for static and global variables

Short name	Structures/SGVariablesConfusion
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.563 Static Loop

Static loop may be preprocessed.

It looks like the following loops are static : the same code is executed each time, without taking into account loop variables.

```
<?php

// Static loop
$total = 0;
for($i = 0; $i < 10; $i++) {
    $total += $i;
}

// The above loop may be replaced by (with some math help)
$total = 10 * (10  + 1) / 2;

// Non-Static loop (the loop depends on the size of the array)
$n = count($array);
for($i = 0; $i < $n; $i++) {
    $total += $i;
}

?>
```

It is possible to create loops that don't use any blind variables, though this is fairly rare. In particular, calling a method may update an internal pointer, like next() or `SimpleXMLIterator\:\:`next() <https://www.php.net/next>`_`.

It is recommended to turn a static loop into an expression that avoid the loop. For example, replacing the sum of all integers by the `function $n * ($n + 1) / 2`, or using array_sum().

This analysis doesn't detect usage of variables with `compact`.

9.563.1 Suggestions

- Precalculate the result of that loop and removes it altogether
- Check that the loop is not missing a blind variable usage
- Replace the usage of a loop with a native PHP call : for example, with str_repeat(). Although the loop is still here, it usually reflects better the intend.

Short name	Structures/StaticLoop
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)

9.564 Static Methods Called From Object

Static methods may be called without instantiating an object. As such, they never interact with the special variable '$this', as they do not depend on object existence.

Besides this, static methods are normal methods that may be called directly from object context, to perform some utility task.

To maintain code readability, it is recommended to call static method in a static way, rather than within object context.

```
<?php
    class x {
        static function y( ) {}
    }

    $z = new x( );

    $z->y( ); // Readability : no one knows it is a static call
    x::y( );  // Readability : here we know
?>
```

Short name	Classes/StaticMethodsCalledFromObject
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.565 Static Methods Can't Contain $this

Static methods are also called `class methods` : they may be called even if the class has no instantiated object. Thus, the local variable `$this` won't exist, PHP will set it to NULL as usual.

```
<?php

class foo {
    // Static method may access other static methods, or property, or none.
    static function staticBar() {
        // This is not possible in a static method
        return self::otherStaticBar() . static::$staticProperty;
    }

    static function bar() {
        // This is not possible in a static method
        return $this->property;
    }
}

?>
```

Either this is not a static method, which is fixed by removing the `static` keyword, or replace all $this mention by static properties `Class\:\:$property`.

See also Static Keyword <https://www.php.net/manual/en/language.oop5.'static.php>'_

9.565.1 Suggestions

- Remove any $this usage
- Turn any $this usage into a static call : $this->foo() => self::foo()

Short name	Classes/StaticContainsThis
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	no-static-this
Examples	*xataface*, *SugarCrm*

9.566 Strange Name For Constants

Those constants looks like a typo from other names.

```
<?php

// This code looks OK : DIRECTORY_SEPARATOR is a native PHP constant
$path = $path . DIRECTORY_SEPARATOR . $file;

// Strange name DIRECOTRY_SEPARATOR
$path = $path . DIRECOTRY_SEPARATOR . $file;

?>
```

9.566.1 Suggestions

- Fix any typo in the spelling of the constants
- Tell us about common misspelling so we can upgrade this analysis

Short name	Constants/StrangeName
Rulesets	*Analyze*, *Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)

9.567 Strange Name For Variables

Variables with strange names. They might be a typo, or bear strange patterns.

Any variable with three identical letter in a row are considered as strange. 2 letters in a row is classic, and while three letters may happen, it is rare enough.

A list of classic typo is also used to find such variables.

This analysis is case-sensitive.

```
<?php

class foo {
    function bar() {
        // Strange name $tihs
        return $tihs;
    }

    function barbar() {
        // variables with blocks of 3 times the same character are reported
        // Based on Alexandre Joly's tweet
        $aaa = $bab + $www;
    }
}

?>
```

See also #QuandLeDevALaFleme.

9.567.1 Suggestions

- Fix the name of the variable
- Rename the variable to something better
- Drop the variable

Short name	Variables/StrangeName
Rulesets	none
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*FuelCMS*, *PhpIPAM*

9.568 Strict Comparison With Booleans

Strict comparisons prevent from mistaking an error with a false.

Boolean values may be easily mistaken with other values, especially when the function may return integer or boolean as a normal course of action.

It is encouraged to use strict comparison === or !== when booleans are involved in a comparison.

```
<?php

// distinguish between : $b isn't in $a, and, $b is at the beginning of $a
if (strpos($a, $b) === 0) {
    doSomething();
}

// DOES NOT distinguish between : $b isn't in $a, and, $b is at the beginning of $a
if (strpos($a, $b)) {
    doSomething();
}
```

(continues on next page)

(continued from previous page)

```
// will NOT mistake 1 and true
$a = array(0, 1, 2, true);
if (in_array($a, true, true)) {
    doSomething();
}

// will mistake 1 and true
$a = array(0, 1, 2, true);
if (in_array($a, true)) {
    doSomething();
}

?>
```

switch() structures always uses == comparisons.

Native function in_array() has a third parameter to make it use strict comparisons.

9.568.1 Suggestions

- Use strict comparison whenever possible

Short name	Structures/BooleanStrictComparison
Rulesets	*Analyze*, *Suggestions*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Phinx*, *Typo3*

9.569 String Initialization

It used to be possible to initialize a variable with an string, and use it as an array. It is not the case anymore in PHP 7.1.

```
<?php

// Initialize arrays with array()
$a = array();
$a[3] = 4;

// Don't start with a string
$a = '';
$a[3] = 4;
print $a;

// Don't start with a string
if (is_numeric($a)) {
    $a[] = $a;
}

?>
```

See also PHP 7.1 no longer converts string to arrays the first time a value is assigned with square bracket notation.

9.569.1 Suggestions

- Always initialize arrays with an empty array(), not a string.

Short name	Arrays/StringInitialization
Rulesets	*CompatibilityPHP71*
Severity	Minor
Time To Fix	Quick (30 mins)

9.570 String May Hold A Variable

Those strings looks like holding a variable.

Single quotes and Nowdoc syntax may include $ signs that are treated as literals, and not replaced with a variable value.

However, there are some potential variables in those strings, making it possible for an error : the variable was forgotten and will be published as such. It is worth checking the content and make sure those strings are not variables.

```
<?php

$a = 2;

// Explicit variable, but literal effect is needed
echo '$a is '.$a;

// One of the variable has been forgotten
echo '$a is $a';

// $CAD is not a variable, rather a currency unit
$total = 12;
echo $total.' $CAD';

// $CAD is not a variable, rather a currency unit
$total = 12;

// Here, $total has been forgotten
echo <<<'TEXT'
$total $CAD
TEXT;

?>
```

Short name	Type/StringHoldAVariable
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Instant (5 mins)

9.571 Strings With Strange Space

An invisible space may be mistaken for a normal space.

However, PHP does straight comparisons, and may fail at recognizing. This analysis reports when it finds such strange spaces inside strings.

PHP doesn't mistake space and tables for whitespace when tokenizing the code.

This analysis doesn't report Unicode Codepoint Notation : those are visible in the code.

```
<?php

// PHP 7 notation,
$a = \u{3000};
$b = ;

// Displays false
var_dump($a === $b);

?>
```

See also Unicode spaces, and disallow irregular whitespace (no-irregular-whitespace).

9.571.1 Suggestions

- Replace the odd spaces with a normal space
- If unsecable spaces are important for presentation, add them at the templating level.

Short name	Type/StringWithStrangeSpace
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*OpenEMR*, *Thelia*

9.572 Strpos()-like Comparison

The result of that function may be mistaken with an error.

strpos(), along with several PHP native functions, returns a string position, starting at 0, or false, in case of failure.

```
<?php

// This is the best comparison
if (strpos($string, 'a') === false) { }

// This is OK, as 2 won't be mistaken with false
if (strpos($string, 'a') == 2) { }

// strpos is one of the 26 functions that may behave this way
if (preg_match($regex, $string)) { }

// This works like above, catching the value for later reuse
if ($a = strpos($string, 'a')) { }

// This misses the case where 'a' is the first char of the string
if (strpos($string, 'a')) { }
```

(continues on next page)

(continued from previous page)

```
// This misses the case where 'a' is the first char of the string, just like above
if (strpos($string, 'a') == 0) { }

?>
```

It is recommended to check the result of strpos() with === or !==, so as to avoid confusing 0 and false.

This analyzer list all the strpos()-like functions that are directly compared with == or !=. preg_match(), when its first argument is a literal, is omitted : this function only returns NULL in case of regex error.

The full list is the following :

- array_search()
- collator_compare()
- collator_get_sort_key()
- current()
- fgetc()
- file_get_contents()
- file_put_contents()
- fread()
- iconv_strpos()
- iconv_strrpos()
- imagecolorallocate()
- imagecolorallocatealpha()
- mb_strlen()
- next()
- pcntl_getpriority()
- preg_match()
- prev()
- readdir()
- stripos()
- strpos()
- strripos()
- strrpos()
- strtok()
- curl_exec()

In PHP 8.0, str_contains() will do the expected job of strpos(), with less confusion.

See also strpos not working correctly.

9.572.1 Suggestions

- Use identity comparisons, for 0 values : === instead of ==, etc.
- Compare with other exact values than 0 : strpos() == 2
- Use str_contains()

Short name	Structures/StrposCompare
Rulesets	*Analyze*, *Top10*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	strict-comparisons
Examples	*Piwigo*, *Thelia*

9.573 Strtr Arguments

Strtr() replaces characters by others in a string. When using strings, strtr() replaces characters as long as they have a replacement. All others are ignored.

In particular, strtr() works on strings of the same size, and cannot be used to remove chars.

```
<?php

$string = 'abcde';
echo strtr($string, 'abc', 'AB');
echo strtr($string, 'ab', 'ABC');
// displays ABcde
// c is ignored each time

// strtr can't remove a char
echo strtr($string, 'a', '');
// displays a

?>
```

See also strtr.

9.573.1 Suggestions

- Check the call to strtr() and make sure the arguments are of the same size
- Replace strtr() with str_replace(), which works with strings and array, not chars
- Replace strtr() with preg_match(), which works with patterns and not chars

Short name	Php/StrtrArguments
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)
Examples	*SuiteCrm*

9.574 Substr To Trim

When removing the first or the last character of a string, trim() does a more readable job.

trim(), ltrim() and rtrim() accept a string as second argument. Those will all be removed from the endings of the string.

```php
<?php

$a = '$drop the dollar';
$b = substr($a, 1); // drop the first char
$b = ltrim($a, '$'); // remove the initial '$'s


$b = substr($a, 1);     // replace with ltrim()

$b = substr($a, 0, -1); // replace with rtrim()

$b = substr($a, 1, -1); // replace with trim()

?>
```

trim() will remove all occurrences of the requested char(). This may remove a loop with substr(), or remove more than is needed.

trim() doesn't work with multi-bytes strings, but so does substr(). For that, use mb_substr(), as there isn't any mb_trim function (yet).

See also trim, ltrim, rtrim.

9.574.1 Suggestions

- Replace substr() with trim(), ltrim() or rtrim().

Short name	Structures/SubstrToTrim
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.575 Substring First

Always start by reducing a string before applying some transformation on it. The shorter string will be processed faster.

```php
<?php

// fast version
$result = strtolower(substr($string, $offset, $length));

// slower version
$result = substr(strtolower($string), $offset, $length);
?>
```

The gain produced here is greater with longer strings, or greater reductions. They may also be used in loops. This is a micro-optimisation when used on short strings and single string reductions.

This works with any reduction function instead of substr(), like trim(), iconv(), etc.

9.575.1 Suggestions

- Always reduce the string first, then apply some transformation

Short name	Performances/SubstrFirst
Rulesets	*Performances*, *Suggestions*, *Top10*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*SPIP*, *PrestaShop*

9.576 Suspicious Comparison

The comparison seems to be misplaced.

A comparison happens in the last argument, while the actual function expect another type : this may be the case of a badly placed parenthesis.

```php
<?php

// trim expect a string, a boolean is given.
if (trim($str === '')){

}

// Just move the first closing parenthesis to give back its actual meaning
if (trim($str) === ''){

}

?>
```

Original idea by Vladimir Reznichenko.

9.576.1 Suggestions

- Remove the comparison altogether
- Move the comparison to its right place : that, or more the parenthesis.
- This may be what is intended : just leave it.

Short name	Structures/SuspiciousComparison
Rulesets	*Analyze*
Severity	Major
Time To Fix	Instant (5 mins)
Examples	*PhpIPAM*, *ExpressionEngine*

9.577 Swapped Arguments

Overwritten methods must be compatible, but argument names is not part of that compatibility.

Methods with the same name, in two classes of the same hierarchy, must be compatible for typehint, default value, reference. The name of the argument is not taken into account when checking such compatibility, at least until PHP 7.4.

```
<?php

class x {
    function foo($a, $b) {}

    function bar($a, $b) {}
}

class y extends x {
    // foo is compatible (identical) with the above class
    function foo($a, $b) {}

    // bar is compatible with the above class, yet, the argument might not receive␣
↪what they expect.
    function bar($b, $a) {}
}

?>
```

This analysis reports argument lists that differs in ordering. This analysis doesn't report argument lists that also differs in argument names.

9.577.1 Suggestions

- Make sure the names of the argument are in the same order in all classes and interfaces

Short name	Classes/SwappedArguments
Rulesets	*Analyze*
Severity	Critical
Time To Fix	Quick (30 mins)

9.578 Switch Fallthrough

A switch with fallthrough is prone to errors.

A fallthrough happens when a case or default clause in a switch statement is not finished by a break (or equivalent); CWE report this as a security concern, unless well documented.

A fallthrough may be used as a feature. Then, it is indistinguishable from an error.

When the case block is empty, this analysis doesn't report it : the case is then used as an alias.

```
<?php
switch($variable) {
    case 1 :   // case 1 is not reported, as it actually shares the same body as case␣
↪33
```

(continues on next page)

(continued from previous page)

```
    case 33 :
        break ;
    case 2 :
        break ;
    default:
        ++$a;
    case 4 :
        break ;
}
?>
```

This analysis doesn't take into account comments about the fallthrough.

See also CWE-484: Omitted 'Break Statement in Switch <https://cwe.mitre.org/data/definitions/484.html>'_ and Rule: no-switch-case-fall-through.

9.578.1 Suggestions

- Make separate code for each case. Always use break at the end of a case or default.

Short name	Structures/Fallthrough
Rulesets	*Security*
Severity	Minor
Time To Fix	Instant (5 mins)

9.579 Switch To Switch

The following structures are based on if / elseif / else. Since they have more than three conditions (not withstanding the final else), it is recommended to use the switch structure, so as to make this more readable.

On the other hand, switch() structures with less than 3 elements should be expressed as a if / else structure.

Note that if condition that uses strict typing (=== or !==) can't be converted to switch() as the latter only performs == or != comparisons.

```
<?php

if ($a == 1) {

} elseif ($a == 2) {

} elseif ($a == 3) {

} elseif ($a == 4) {

} else {

}

// Better way to write long if/else lists
switch ($a) {
    case 1 :
        doSomething(1);
```

(continues on next page)

(continued from previous page)

```
        break 1;

    case 2 :
        doSomething(2);
        break 1;

    case 3 :
        doSomething(3);
        break 1;

    case 4 :
        doSomething(4);
        break 1;

    default :
        doSomething();
        break 1;
}

?>
```

Note that simple switch statement, which compare a variable to a literal are optimised in PHP 7.2 and more recent. This gives a nice performance boost, and keep code readable.

See also PHP 7.2's switch optimisations and Is Your Code Readable By Humans? Cognitive Complexity Tells You.

9.579.1 Suggestions

- Use a switch statement, rather than a long string of if/else
- Use a match() statement, rather than a long string of if/else (PHP 8.0 +)

Short name	Structures/SwitchToSwitch
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Thelia*, *XOOPS*

9.580 Switch With Too Many Default

Switch statements should only hold one default, not more. Check the code and remove the extra default.

PHP 7.0 won't compile a script that allows for several default cases.

Multiple default happens often with large switch().

```
<?php

switch($a) {
    case 1 :
        break;
    default :
        break;
```

(continues on next page)

(continued from previous page)

```
    case 2 :
        break;
    default :  // This default is never reached
        break;
}

?>
```

9.580.1 Suggestions

- Remove the useless default : it may be the first, or the last. In case of ambiguity, keep the first, as it is the one being used at the moment.

Short name	Structures/SwitchWithMultipleDefault
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and older
Severity	Minor
Time To Fix	Quick (30 mins)

9.581 Switch Without Default

Always use a default statement in switch().

Switch statements hold a number of ‘case’ that cover all known situations, and a ‘default’ one which is executed when all other options are exhausted.

```
<?php

// Missing default
switch($format) {
    case 'gif' :
        processGif();
        break 1;

    case 'jpeg' :
        processJpeg();
        break 1;

    case 'bmp' :
        throw new UnsupportedFormat($format);
}
// In case $format is not known, then switch is ignored and no processing happens,␣
↪leading to preparation errors


// switch with default
switch($format) {
    case 'text' :
        processText();
        break 1;

    case 'jpeg' :
```

(continues on next page)

(continued from previous page)

```
        processJpeg();
        break 1;

    case 'rtf' :
        throw new UnsupportedFormat($format);

    default :
        throw new UnknownFileFormat($format);
}
// In case $format is not known, an exception is thrown for processing

?>
```

Most of the time, switch() do need a default case, so as to catch the odd situation where the ‘value is not what it was expected’. This is a good place to catch unexpected values, to set a default behavior.

9.581.1 Suggestions

- Add a default case

Short name	Structures/SwitchWithoutDefault
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	no-switch-without-default
Examples	*Zencart*, *Traq*

9.582 Ternary In Concat

Ternary and coalesce operator have higher priority than dot ‘.’ for concatenation. This means that :

```
<?php
  // print B0CE as expected
  print 'B'.$b.'C'. ($b > 1 ? 'D') : 'E';

  // print E, instead of B0CE
  print 'B'.$b.'C'. $b > 1 ? 'D' : 'E';

  print 'B'.$b.'C'. $b > 1 ? 'D' : 'E';
?>
```

prints actually ‘E’, instead of the awaited ‘B0CE’.

To be safe, always add parenthesis when using ternary operator with concatenation.

See also Operator Precedence.

9.582.1 Suggestions

- Use parenthesis
- Avoid ternaries and coalesce operators inside a string

Short name	Structures/TernaryInConcat
Rulesets	*Analyze*, *CI-checks*
Severity	Critical
Time To Fix	Quick (30 mins)
Examples	*TeamPass*

9.583 Test Then Cast

A test is run on the value, but the cast value is later used.

The cast may introduce a distortion to the value, and still lead to the unwanted situation. For example, comparing to 0, then later casting to an int. The comparison to 0 is done without casting, and as such, 0.1 is different from 0. Yet, (int) 0.1 is actually 0, leading to a Division by 0 error.

```php
<?php

// Here. $x may be different from 0, but (int) $x may be 0
$x = 0.1;

if ($x != 0) {
    $y = 4 / (int) $x;
}

// Safe solution : check the cast value.
if ( (int) $x != 0) {
    $y = 4 / (int) $x;
}

?>
```

9.583.1 Suggestions

- Test with the cast value

Short name	Structures/TestThenCast
Rulesets	*Analyze*
Severity	Major
Time To Fix	Instant (5 mins)
Examples	*Dolphin*, *SuiteCrm*

9.584 Throw Functioncall

The `throw` keyword expects to use an exception. Calling a function to prepare that exception before throwing it is possible, but forgetting the new keyword is also possible.

```php
<?php

// Forgotten new
throw \RuntimeException('error!');
```

(continues on next page)

(continued from previous page)

```
// Code is OK, function returns an exception
throw getException(ERROR_TYPE, 'error!');

function getException(ERROR_TYPE, $message) {
    return new \RuntimeException($messsage);
}

?>
```

When the `new` keyword is forgotten, then the class constructor is used as a function name, and now exception is emitted, but an `Undefined function` fatal error is emitted.

See also Exceptions.

9.584.1 Suggestions

- Add the new operator to the call
- Make sure the function is really a functioncall, not a class name

Short name	Exceptions/ThrowFunctioncall
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)
Examples	*SugarCrm*, *Zurmo*

9.585 Throw In Destruct

According to the manual, `Attempting to throw an exception from a destructor (called in the time of script termination) causes a fatal error.`

The destructor may be called during the lifespan of the script, but it is not certain. If the exception is thrown later, the script may end up with a fatal error.

Thus, it is recommended to avoid throwing exceptions within the `__destruct` method of a class.

```
<?php

// No exception thrown
class Bar {
    function __construct() {
        throw new Exception('__construct');
    }

    function __destruct() {
        $this->cleanObject();
    }
}

// Potential crash
class Foo {
    function __destruct() {
```

(continues on next page)

(continued from previous page)

```
        throw new Exception('__destruct');
    }
}

?>
```

See also Constructors and Destructors.

9.585.1 Suggestions

- Remove any exception thrown from a destructor

Short name	Classes/ThrowInDestruct
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)

9.586 Throw Was An Expression

Throw used to be an expression. In PHP 7.0, there were some location where one couldn't use a throw : this was the case for arrow functions, which expect one expression as function's body.

Using throw as an instruction makes the code incompatible with PHP 7 version and older.

```
<?php

// Valid in PHP 8.0 and more recent
$fn = fn($a) => throw new Exception($a);

?>
```

See also Throw Expression and Exceptions.

9.586.1 Suggestions

-

Short name	Php/ThrowWasAnExpression
Rulesets	*CompatibilityPHP72*, *CompatibilityPHP73*, *CompatibilityPHP74*
Php Version	8.0+
Severity	Major
Time To Fix	Quick (30 mins)

9.587 Throws An Assignement

It is possible to throw an exception, and, in the same time, assign this exception to a variable.

However, the variable will never be used, as the exception is thrown, and any following code is not executed, unless the exception is caught in the same scope.

```php
<?php

    // $e is useful, though not by much
    $e = new() Exception();
    throw $e;

    // $e is useless
    throw $e = new Exception();

?>
```

9.587.1 Suggestions

- Drop the assignation

Short name	Structures/ThrowsAndAssign
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)

9.588 Timestamp Difference

`time()` and `microtime()` shouldn’t be used to calculate duration.

`time()` and `microtime()` are subject to variations, depending on system clock variations, such as daylight saving time difference (every spring and fall, one hour variation), or leap seconds, happening on `June, 30th` or `December 31th`, as announced by IERS.

```php
<?php

// Calculating tomorow, same hour, the wrong way
// tomorrow is not always in 86400s, especially in countries with daylight saving
$tomorrow = time()  + 86400;

// Good way to calculate tomorrow
$datetime = new DateTime('tomorrow');

?>
```

When the difference may be rounded to a larger time unit (rounding the difference to days, or several hours), the variation may be ignored safely.

When the difference is very small, it requires a better way to measure time difference, such as *Ticks <https://www.php.net/manual/en/control-structures.declare.php#control-structures.declare.ticks>’_, ‘ext/hrtime <https://www.php.net/manual/en/book.hrtime.php>’_, or including a check on the actual time zone (‘‘ini_get()‘* with ‘date.timezone’).

See also PHP DateTime difference – it’s a trap! and PHP Daylight savings bug?.

9.588.1 Suggestions

- For small time intervals, use hrtime() functions

- For larger time intervals, use add() method with `DateTime`

Short name	Structures/TimestampDifference
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Slow (1 hour)
Examples	*Zurmo*, *shopware*

9.589 Too Long A Block

The loop is operating on a block that is too long.

This analysis is applied to loops (for, foreach, while, do..while) and if/then/else/elseif structures.

Then length of a block is managed with the ‘‘longBlock‘‘parameter. By default, it is 200 lines, from beginning to the end. Comments are taken into account.

```
<?php

$i = 0;
do {
    // 200 lines of PHP code

    ++$i;
} while($i < 100);

?>
```

9.589.1 Suggestions

- Move the code of the block to an method or a function
- Move part of the code of the block to methods or functions
- Extract repeated patterns and use them

Name	Default	Type	Description
longBlock	200	integer	Size of a block for it to be too long. A block is commanded by a for, foreach, while, do…while, if/then else structure.

Short name	Structures/LongBlock
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.590 Too Many Array Dimensions

When arrays a getting to many nesting.

```
<?php

$a           = array();    // level 1;
$a[1]        = array();    // level 2
$a[1][2]     = array();    // level 3 : still valid by default
$a[1][2][3] = array();    // level 4

?>
```

PHP has no limit, and accepts any number of nesting levels. Yet, this is usually very memory hungry.

9.590.1 Suggestions

-

Name	Default	Type	Description
maxDimensions	3	integer	Number of valid dimensions in an array.

Short name	Arrays/TooManyDimensions
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.591 Too Many Children

Classes that have more than 15 children. It is worth checking if they cannot be refactored in anyway.

The threshold of 15 children can be configured. There is no technical limitation of the number of children and grandchildren for a class.

The analysis doesn't work recursively : only direct generations are counted. Only children that can be found in the code are counted.

```
<?php

// parent class
// calling it grandparent to avoid confusion with 'parent'
class grandparent {}


class children1 extends grandparent {}
class children2 extends grandparent {}
class children3 extends grandparent {}
class children4 extends grandparent {}
class children5 extends grandparent {}
class children6 extends grandparent {}
class children7 extends grandparent {}
class children8 extends grandparent {}
class children9 extends grandparent {}
class children11 extends grandparent {}
class children12 extends grandparent {}
class children13 extends grandparent {}
```

(continues on next page)

(continued from previous page)

```
class children14 extends grandparent {}
class children15 extends grandparent {}
class children16 extends grandparent {}
class children17 extends grandparent {}
class children18 extends grandparent {}
class children19 extends grandparent {}

?>
```

See also Why is subclassing too much bad (and hence why should we use prototypes to do away with it)?.

9.591.1 Suggestions

- Split the original class into more specialised classes

Name	Default	Type	Description
childrenClassCount	15	integer	Threshold for too many children classes for one class.

Short name	Classes/TooManyChildren
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*Typo3*, *Woocommerce*

9.592 Too Many Dereferencing

Linking too many properties and methods, one to the other.

This analysis counts both static calls and normal call; methods, properties and constants. It also takes into account arrays along the way.

The default limit of chaining methods and properties is set to 7 by default.

```
<?php

// 9 chained calls.
$main->getA()->getB()->getC()->getD()->getE()->getF()->getG()->getH()->getI()->
↪property;

?>
```

Too many chained methods is harder to read.

9.592.1 Suggestions

-

Name	Default	Type	Description
tooManyDereferencing	7	integer	Maximum number of dereferencing.

Short name	Classes/TooManyDereferencing
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.593 Too Many Finds

Too many methods called 'find*' in this class. It is may be time to consider the Specification pattern.

```
<?php

// quite a fishy interface
interface UserInterface {
    public function findByEmail($email);
    public function findByUsername($username);
    public function findByFirstName($firstname);
    public function findByLastName($lastname);
    public function findByName($name);
    public function findById($id);

    public function insert($user);
    public function update($user);
}

?>
```

See also On Taming Repository Classes in Doctrine , On Taming Repository Classes in Doctrine… Among other things., specifications.

Name	Default	Type	Description
minimumFinds	5	integer	Minimal number of prefixed methods to report.
findPrefix	find	string	list of prefix to use when detecting the 'find'. Comma-separated list, case insensitive.
findSuffix		string	list of fix to use when detecting the 'find'. Comma-separated list, case insensitive.

Short name	Classes/TooManyFinds
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)

9.594 Too Many Injections

When a class is constructed with more than four dependencies, it should be split into smaller classes.

```
<?php
```

(continues on next page)

(continued from previous page)

```
// This class relies on 5 other instances.
// It is probably doing too much.
class Foo {
    public function __construct(
            A $a,
            B $b,
            C $c,
            D $d
            E $e ) {
        $this->a = $a;
        $this->b = $b;
        $this->d = $d;
        $this->d = $d;
        $this->e = $e;
    }
}

?>
```

See also Dependency Injection Smells.

9.594.1 Suggestions

- Split the class into smaller classes. Try to do less in that class.

Name	Default	Type	Description
injectionsCount	5	integer	Threshold for too many injected parameters for one class.

Short name	Classes/TooManyInjections
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*NextCloud*, *Thelia*

9.595 Too Many Local Variables

Too many local variables were found in the methods. When over 15 variables are found in such a method, a violation is reported.

Local variables exclude globals (imported with global) and arguments. Local variable include static variables.

When too many variables are used in a function, it is a code smells. The function is trying to do too much and needs extra space for juggling. Beyond 15 variables, it becomes difficult to keep track of their name and usage, leading to confusion, overwriting or hijacking.

```
<?php

// This function is OK : 3 vars are arguments, 3 others are globals.
function a20a3g3($a1, $a2, $a3) {
    global $a4, $a5, $a6;
```

(continues on next page)

(continued from previous page)

```
    $a1  = 1;
    $a2  = 2;
    $a3  = 3 ;
    $a4  = 4 ;
    $a5  = 5 ;
    $a6  = 6 ;
    $a7  = 7 ;
    $a8  = 8 ;
    $a9  = 9 ;
    $a10 = 10;
    $a11  = 11;
    $a12  = 12;
    $a13  = 13 ;
    $a14  = 14 ;
    $a15  = 15 ;
    $a16  = 16 ;
    $a17  = 17 ;
    $a18  = 18 ;
    $a19  = 19 ;
    $a20 = 20;

}

// This function has too many variables
function a20() {

    $a1  = 1;
    $a2  = 2;
    $a3  = 3 ;
    $a4  = 4 ;
    $a5  = 5 ;
    $a6  = 6 ;
    $a7  = 7 ;
    $a8  = 8 ;
    $a9  = 9 ;
    $a10 = 10;
    $a11  = 11;
    $a12  = 12;
    $a13  = 13 ;
    $a14  = 14 ;
    $a15  = 15 ;
    $a16  = 16 ;
    $a17  = 17 ;
    $a18  = 18 ;
    $a19  = 19 ;
    $a20 = 20;

}

?>
```

9.595.1 Suggestions

- Remove some of the variables, and inline them
- Break the big function into smaller ones

- Find repeated code and make it a separate function

Name	Default	Type	Description
tooManyLocalVariableThreshold	15	integer	Minimal number of variables in one function or method to report.

Short name	Functions/TooManyLocalVariables
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*HuMo-Gen*

9.596 Too Many Native Calls

Avoid stuffing too many PHP native call inside another functioncall.

For readability reasons, or, more often, for edge case handling, it is recommended to avoid nesting too many PHP native calls.

This analysis reports any situation where more than 3 PHP native calls are nested.

```
<?php

// Too many nested functions
$cleanArray = array_unique(array_keys(array_count_values(array_column($source, 'x
↪')))));

// Avoid warning when source is empty
$extract = array_column($source, 'x');
if (empty($extract)) {
    $cleanArray = array();
} else {
    $cleanArray = array_unique(array_keys(array_count_values($extract)));
}

// This is not readable, although it is short.
// It may easily get out of hand.
echo chr(80), chr(72), chr(80), chr(32), ' is great!';

?>
```

Name	Default	Type	Description
nativeCallCounts	3	integer	Number of native calls found inside another call.

Short name	Php/TooManyNativeCalls
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*SPIP*

9.597 Too Many Parameters

Method has too many parameters. Exakat has a default parameter count which may be configured.

A method that needs more than 8 parameters is trying to do too much : it should be reviewed and split into smaller methods.

```
<?php

// This methods has too many parameters.
function alertSomeone($name, $email, $title, $message, $attachements, $signature,
↪$bcc, $cc, $extra_headers) {
    /* too much code here */
}

?>
```

See also How many parameters is too many ? and Too Many Parameters.

9.597.1 Suggestions

- Reduce the number of parameters to a lower level
- Break the function into smaller functions
- Turn the function into a class

Name	Default	Type	Description
parametersCount	8	integer	Minimal number of parameters to report.

Short name	Functions/TooManyParameters
Rulesets	*Suggestions*
Examples	*WordPress*, *ChurchCRM*

9.598 Too Much Indented

Reports methods that are using more than one level of indentation on average.

Indentations levels are counted for each for, foreach, if. . . then, while, do..while, try..catch..finally structure met. Compulsory expressions, such as conditions, are not counted in the total. Levels of indentation start at 0 (no indentation needed)

This analysis targets methods which are build around large conditions : the actual useful code is nested inside the branches of the if/then/else (for example).

The default threshold `indentationAverage` of 1 is a good start for spotting large methods with big conditional code, and will leave smaller methods, even when they only contain one if/then. Larger methods shall be refactored in smaller size.

The parameter `minimumSize` set aside methods which are too small for refactoring.

```
<?php

// average 0
function foo0() {
    $a = rand(1,2);
    $a *= 3;

    return $a;
}

// average 0.66 = (0 + 1 + 1) / 3
function foo0_66() {
    // if () is at level 0
    if ($a == 2) { // condition is not counted
        $a = 1;    // level 1
    } else {
        $a = 2;    // level 1
    }
}

// average 1 = (0 + 2 + 1 + 1) / 4
function foo1() {
    // if () is at level 0
    if ($a == 2) {
        // if () is at level 1
        if ($a == 2) {
            $a = 1; // level 2
        }
        $a = 1; // level 1
    } else {
        $a = 2; // level 1
    }
}

?>
```

This analysis is distinct from Structures/MaxLevelOfIdentation, which only reports the highest level of indentation. This one reports how one method is build around one big

See also *Max Level Of Nesting*.

9.598.1 Suggestions

- Refactor the method to reduce the highest level of indentation
- Refactor the method move some of the code to external methods.

Name	Default	Type	Description
indentationAverage	1	real	Minimal average of indentation in a method to report. Default is 1.0, which means that the method is on average at one level of indentation or more.
minimumSize	3	real	Minimal number of expressions in a method to apply this analysis.

Short name	Functions/TooMuchIndented
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.599 Trailing Comma In Calls

The last argument may be left empty.

This feature was introduced in PHP 7.3.

```
<?php

// VCS friendly call
// PHP 7.3 and more recent
foo(1,
    2,
    3,
   );

// backward compatible call
// All PHP versions
foo(1,
    2,
    3
   );

?>
```

See also PHP RFC: Allow a trailing comma in function calls.

Short name	Php/TrailingComma
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Severity	Minor
Time To Fix	Quick (30 mins)

9.600 Trait Not Found

A unknown trait is mentioned in the use expression.

The used traits all exist, but in the configuration block, some unmentioned trait is called.

Be aware that the traits used in any configuration block may originate in any use expression. PHP will check the configuration block at instantiation only, and after compiling : at that moment, it will know all the used traits across the class.

```
<?php
class x  {
    // c is not a used trait
```

(continues on next page)

(continued from previous page)

```
    use a, b { c::d insteadof e;}

    // e is a used trait, even if is not in the use above.
    use e;
}
?>
```

See also Traits.

9.600.1 Suggestions

- Switch the name of the trait to an existing and used trait
- Drop the expression that rely on the non-existent trait

Short name	Traits/TraitNotFound
Rulesets	*Analyze*, *LintButWontExec*
Severity	Minor
Time To Fix	Quick (30 mins)

9.601 Typed Property Usage

Traditionally, PHP properties aren't typed. Since PHP 7.4, it is possible to type properties, just like arguments.

```
<?php

class User {
    public int $id;
    public string $name;

    public function __construct(int $id, string $name) {
        $this->id = $id;
        $this->name = $name;
    }
}
?>
```

See also Typed Properties 2.0.

9.601.1 Suggestions

-

Short name	Php/TypedPropertyUsage
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)

9.602 Typehint Must Be Returned

When using a typehint for a method, it is compulsory to use a at least one return in the method's body. This is true for nullable typehint too : `return` alone won't be sufficient.

```
<?php

// The function returns a value (here, correct object)
function foo() : Bar { return new Bar(); }

// The function should at least, return a value
function foo() : Bar { }

// The function should at least, return a value : Null or an object. Void, here, is
↪not acceptable.
function foo() : ?Bar { return; }

?>
```

PHP lint this, but won't execute it.

This analysis doesn't check if the returned value is compatible with the returned typehint. Only its presence is checked.

See also Return Type Declaration and Type hint in PHP function parameters and return values.

9.602.1 Suggestions

- Add a return with a valid value

Short name	Functions/TypehintMustBeReturned
Rulesets	*Analyze*, *LintButWontExec*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)

9.603 Typehinted References

Typehinted arguments have no need for references. Since they are only an object, they are already a reference.

In fact, adding the & on the argument definition may lead to error like `Only variables should be passed by reference`.

This applies to the `object` type hint, but not the the others, such as `int` or `bool`.

```php
<?php
    // a class
    class X {
        public $a = 3;
    }

    // typehinted reference
    //function foo(object &$x) works too
    function foo(X &$x) {
        $x->a = 1;

        return $x;
    }

    // Send an object
    $y = foo(new X);

    // This prints 1;
    print $y->a;
?>
```

See also Passing by reference and Objects and references.

9.603.1 Suggestions

- Remove reference for typehinted arguments, unless the typehint is a scalar typehint.

Short name	Functions/TypehintedReferences
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

9.604 Unbinding Closures

Never drop `$this`, once a closure was created in a non-static method.

From the PHP wiki : Currently it is possible to unbind the $this variable from a closure that originally had one by using $closure->bindTo(null). Due to the removal of static calls to non-static methods in PHP 8, we now have a guarantee that $this always exists inside non-static methods. We would like to have a similar guarantee that $this always exists for non-static closures declared inside non-static methods. Otherwise, we will end up imposing an unnecessary performance penalty either on $this accesses in general, or $this accesses inside such closures.

```php
<?php

class x {
    private $a = 3;

```

(continues on next page)

(continued from previous page)

```
    function foo() {
        return function () { echo $this->a; };
    }
}

$closure = (new x)->foo();

// $this was expected, and it is not anymore
$closure->bindTo(null);

$closure->bindTo(new x);

?>
```

Calling bindTo() with a valid object is still valid.

See also Unbinding ‘$this from non-static closures <https://wiki.php.net/rfc/deprecations_php_7_4#unbinding_this_from_non-static_closures>‘_.

9.604.1 Suggestions

- Create a static closure, which doesn’t rely on $this at all
- Remove the call to bindTo(null).

Short name	Functions/UnbindingClosures
Rulesets	*CompatibilityPHP74*
Severity	Minor
Time To Fix	Quick (30 mins)

9.605 Uncaught Exceptions

The following exceptions are thrown in the code, but are never caught.

```
<?php

// This exception is throw, but not caught. It will lead to a fatal error.
if ($message = check_for_error()) {
    throw new My\Exception($message);
}

// This exception is throw, and caught.
try {
    if ($message = check_for_error()) {
        throw new My\Exception($message);
    }
} catch (\Exception $e) {
    doSomething();
}

?>
```

Either they will lead to a Fatal Error, or they have to be caught by an including application. This is a valid behavior for libraries, but is not for a final application.

See also Structuring PHP Exceptions.

9.605.1 Suggestions

- Catch all the exceptions you throw

Short name	Exceptions/UncaughtExceptions
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.606 Unchecked Resources

Resources are created, but never checked before being used. This is not safe.

Always check that resources are correctly created before using them.

```php
<?php

// always check that the resource is created correctly
$fp = fopen($d,'r');
if ($fp === false) {
    throw new Exception('File not found');
}
$firstLine = fread($fp);

// This directory is not checked : the path may not exist and return false
$uncheckedDir = opendir($pathToDir);
while(readdir($uncheckedDir)) {
    // do something()
}

// This file is not checked : the path may not exist or be unreadable and return false
$fp = fopen($pathToFile);
while($line = freads($fp)) {
    $text .= $line;
}

// unsafe one-liner : using bzclose on an unchecked resource
bzclose(bzopen('file'));

?>
```

See also resources.

Short name	Structures/UncheckedResources
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Slow (1 hour)
ClearPHP	no-unchecked-resources

9.607 Unconditional Break In Loop

An unconditional break in a loop creates dead code. Since the break is directly in the body of the loop, it is always executed, creating a strange loop that can only run once.

Here, break may also be a return, a goto or a continue. They all branch out of the loop. Such statement are valid, but should be moderated with a condition.

```
<?php

// return in loop should be in
function summAll($array) {
    $sum = 0;

    foreach($array as $a) {
        // Stop at the first error
        if (is_string($a)) {
            return $sum;
        }
        $sum += $a;
    }

    return $sum;
}

// foreach loop used to collect first element in array
function getFirst($array) {
    foreach($array as $a) {
        return $a;
    }
}

?>
```

9.607.1 Suggestions

- Remove the loop and call the content of the loop once.

Short name	Structures/UnconditionLoopBreak
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*LiveZilla*, *MediaWiki*

9.608 Undefined ::class

`\:\:class` doesn't check if a corresponding class exists.

`\:\:class` must be checked with a call to class_exists(). Otherwise, it may lead to a `Class 'foo' not found` or even silent dead code : this happens also with Catch and instanceof commands with undefined classes. PHP doesn't raise an error in that case.

```
<?php

class foo() {}

// prints foo
echo foo::class;

// prints bar though bar doesn't exist.
echo bar::class;

?>
```

See also Class Constants.

Short name	Classes/UndefinedStaticclass
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)

9.609 Undefined Caught Exceptions

Those are exceptions that are caught in the code, but are not defined in the application.

They may be externally defined, such as in core PHP, extensions or libraries. Make sure those exceptions are useful to your application : otherwise, they are dead code.

```
<?php

try {
    library_function($some, $args);

} catch (LibraryException $e) {
    // This exception is not defined, and probably belongs to Library
    print Library failed\n;

} catch (OtherLibraryException $e) {
    // This exception is not defined, and probably do not belongs to this code
    print Library failed\n;

} catch (\Exception $e) {
    // This exception is a PHP standard exception
    print Something went wrong, but not at Libary level\n;
}

?>
```

9.609.1 Suggestions

- Remove the catch clause, as it is dead code
- Make sure the exception is thrown by the underlying code

Short name	Exceptions/CaughtButNotThrown
Rulesets	*Dead code*

9.610 Undefined Class Constants

Class constants that are used, but never defined. This should yield a fatal error upon execution, but no feedback at compile level.

```php
<?php

class foo {
    const A = 1;
    define('B', 2);
}

// here, C is not defined in the code and is reported
echo foo::A.foo::B.foo::C;

?>
```

9.610.1 Suggestions

- Fix the name of the constant
- Add the constant to the current class or one of its parent
- Update the constant's visibility

Short name	Classes/UndefinedConstants
Rulesets	none
Severity	Major
Time To Fix	Quick (30 mins)

9.611 Undefined Classes

Those classes are used in the code, but there are no definition for them.

This may happens under normal conditions, if the application makes use of an unsupported extension, that defines extra classes; or if some external libraries, such as PEAR, are not provided during the analysis.

```php
<?php

// FPDF is a classic PDF class, that is usually omitted by Exakat.
$o = new FPDF();

// Exakat reports undefined classes in instanceof
// PHP ignores them
if ($o instanceof SomeClass) {
    // doSomething();
}
```

(continues on next page)

(continued from previous page)

```

// Classes may be used in typehint too
function foo(TypeHintClass $x) {
    // doSomething();
}

?>
```

Short name	Classes/UndefinedClasses
Rulesets	*Analyze*
Severity	Major
Time To Fix	Slow (1 hour)

9.612 Undefined Constant Name

When using the ‘‘ syntax for variable, the name used must be a defined constant. It is not a simple string, like ‘x’, it is an actual constant name.

Interestingly, it is possible to use a qualified name within ‘‘, full or partial. PHP will lint such code, and will collect the value of the constant immediately. Since there is no fallback mechanism for fully qualified names, this ends with a Fatal error.

```
<?php

const x = a;
$a = Hello;

// Display 'Hello'  -> $a -> Hello
echo ;

// Yield a PHP Warning
// Use of undefined constant y - assumed 'y' (this will throw an Error in a future␣
↪version of PHP)
echo ;

// Yield a PHP Fatal error as PHP first checks that the constant exists
//Undefined constant 'y'
echo ;
?>
```

9.612.1 Suggestions

- Define the constant
- Turn the dynamic syntax into a normal variable syntax
- Use a fully qualified name (at least one) to turn this syntax into a Fatal error when the constant is not found. This doesn’t fix the problem, but may make it more obvious during the diagnostic.

Short name	Variables/UndefinedConstantName
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.613 Undefined Constants

Constants definition can’t be located.

Those constants are not defined in the code, and will raise errors, or use the fallback mechanism of being treated like a string.

```php
<?php

const A = 1;
define('B', 2);

// here, C is not defined in the code and is reported
echo A.B.C;

?>
```

It is recommended to define them all, or to avoid using them.

See also Constants.

9.613.1 Suggestions

- Define the constant
- Fix the name of the constant
- Fix the namespace of the constant (FQN or use)
- Remove the usage of the constant

Short name	Constants/UndefinedConstants
Rulesets	*Analyze*, *Analyze*, *CompatibilityPHP72*, *CI-checks*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

9.614 Undefined Functions

Some functions are called, but not defined in the code. This means that the functions are probably defined in a missing library, or in an extension. If not, this will yield a Fatal error at execution.

```php
<?php

// Undefined function
foo($a);
```

(continues on next page)

(continued from previous page)

```

// valid function, as it belongs to the ext/yaml extension
$parsed = yaml_parse($yaml);

// This function is not defined in the a\b\c namespace, nor in the global namespace
a\b\c\foo();

?>
```

See also Functions.

9.614.1 Suggestions

- Fix the name of the function in the code
- Remove the functioncall in the code
- Define the function for the code to call it
- Include the correct library in the code source

Short name	Functions/UndefinedFunctions
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

9.615 Undefined Insteadof

`Insteadof` tries to replace a method with another, but it doesn't exists. This happens when the replacing class is refactored, and some of its definition are dropped.

`Insteadof` may replace a non-existing method with an existing one, but not the contrary.

```
<?php

trait A {
    function C (){}
}

trait B {
    function C (){}
}

class Talker {
    use A, B {
        B::C insteadof A;
        B::D insteadof A;
    }
}

new Talker();
?>
```

This error is not linted : it only appears at execution time.

See also Traits.

9.615.1 Suggestions

- Remove the insteadof expression
- Fix the original method and replace it with an existing method

Short name	Traits/UndefinedInsteadof
Rulesets	*Analyze*, *LintButWontExec*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)

9.616 Undefined Interfaces

Some typehints or `instanceof` that are relying on undefined interfaces or classes. They will always return false. Any condition based upon them are dead code.

```
<?php

class var implements undefinedInterface {
    // If undefinedInterface is undefined, this code lints but doesn't run
}

if ($o instanceof undefinedInterface) {
    // This is silent dead code
}

function foo(undefinedInterface $a) {
    // This is dead code
    // it will probably be discovered at execution
}

?>
```

See also Object interfaces, Type declarations, and Instanceof.

9.616.1 Suggestions

- Implement the missing interfaces
- Remove the code governed by the missing interface : the whole method if it is an typehint, the whole if/then if it is a condition.

Short name	Interfaces/UndefinedInterfaces
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*xataface*

9.617 Undefined Parent

List of properties and methods that are accessed using `parent` keyword but are not defined in the parent classes.

This may compile but, eventually yields a fatal error during execution.

```php
<?php

class theParent {
    // No bar() method
    // private bar() method is not accessible to theChild
}

class theChild extends theParent {
    function foo() {
        // bar is defined in theChild, but not theParent
        parent::bar();
    }

    function bar() {

    }
}

?>
```

Note that if the parent is defined using `extends someClass` but `someClass` is not available in the tested code, it will not be reported : it may be in composer, another dependency, or just missing.

See also parent <https://www.php.net/manual/en/keyword.‘parent.php>‘_.

9.617.1 Suggestions

- Remove the usage of the found method
- Add a definition for the method in the appropriate parent
- Fix the name of the method, and replace it with a valid definition
- Change ‘parent’ with ‘self’ if the method is eventually defined in the current class
- Change ‘parent’ with another object, if the method has been defined in another class
- Add the ‘extends’ keyword to the class, to actually have a parent class

Short name	Classes/UndefinedParentMP
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.618 Undefined Properties

List of properties that are not explicitly defined in the class, its parents or traits.

```
<?php

class foo {
    // property definition
    private bar = 2;

    function foofoo() {
        // $this->bar is defined in the class
        // $this->barbar is NOT defined in the class
        return $this->bar + $this->barbar;
    }
}

?>
```

It is possible to spot unidentified properties by using the PHP's magic methods `__get` and `__set`. Even if the class doesn't use magic methods, any call to an undefined property will be directed to those methods, and they can be used as a canary, warning that the code is missing a definition.

```
<?php

trait NoUnefinedProperties {
     function __get($name) {
             assert(false, "Attempt to read the $name property, on the class ".__
↪CLASS__;
     }

     function __set($name, $value) {
             assert(false, "Attempt to read the $name property, on the class ".__
↪CLASS__;
     }
}

?>
```

See also Properties.

9.618.1 Suggestions

- Add an explicit property definition, and give it `null` as a default value : this way, it behaves the same as undefined.

Short name	Classes/UndefinedProperty
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	no-undefined-properties
Examples	*WordPress*, *MediaWiki*

9.619 Undefined Trait

Those are undefined, traits .

When the using class or trait is instantiated, PHP emits a a fatal error.

```
<?php

use Composer/Component/someTrait as externalTrait;

trait t {
    function foo() {}
}

// This class uses trait that are all known
class hasOnlyDefinedTrait {
    use t, externalTrait;
}

// This class uses trait that are unknown
class hasUndefinedTrait {
    use unknownTrait, t, externalTrait;
}
?>
```

Trait which are referenced in a *use* expression are omitted: they are considered part of code that is probably outside the current code, either omitted or in external component.

9.619.1 Suggestions

- Define the missing trait
- Remove usage of the missing trait

Short name	Traits/UndefinedTrait
Rulesets	*Analyze*, *LintButWontExec*, *CI-checks*
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

9.620 Undefined Variable

Variable that is used before any creation.

It is recommended to use a default value for every variable used. When not specified, the default value is set to NULL by PHP.

```
<?php

// Adapted from the PHP manual
$var = 'Bob';
$Var = 'Joe';
// The following line may emit a warning : Undefined variable: $undefined
echo $var, $Var, $undefined;      // outputs Bob, Joe,


?>
```

Variable may be created in various ways : assignation, arguments, foreach blind variables, static and global variables.

This analysis doesn't handle dynamic variables, such as `$$x`. It also doesn't handle variables outside a method or function.

See also Variable basics.

9.620.1 Suggestions

- Remove the expression that is using the undefined variable
- Fix the variable name
- Define the variable by assigning a value to it, before using it

Short name	Variables/UndefinedVariable
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.621 Undefined static:: Or self::

self and static refer to the current class, or one of its parent. The property or the method may be undefined.

```
<?php

class x {
    static public function definedStatic() {}
    private definedStatic = 1;

    public function method() {
        self::definedStatic();
        self::undefinedStatic();

        static::definedStatic;
        static::undefinedStatic;
    }
}

?>
```

See also Late ‘Static Bindings <https://www.php.net/manual/en/language.oop5.late-static-bindings.php>‘_.

9.621.1 Suggestions

- Define the missing method or property
- Remove usage of that undefined method or property
- Fix name to call an actual local structure

Short name	Classes/UndefinedStaticMP
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*xataface*, *SugarCrm*

9.622 Unicode Escape Partial

PHP 7 introduces a new escape sequence for strings : `u{hex}`. It is backward incompatible with previous PHP versions for two reasons :

PHP 7 will recognize en replace those sequences, while PHP 5 keep them intact. PHP 7 will halt on partial Unicode Sequences, as it tries to understand them, but may fail.

```
<?php

echo \u{1F418}\n;
// PHP 5 displays the same string
// PHP 7 displays : an elephant

echo \u{NOT A UNICODE CODEPOINT}\n;
// PHP 5 displays the same string
// PHP 7 emits a fatal error

?>
```

Is is recommended to check all those strings, and make sure they will behave correctly in PHP 7.

Short name	Php/UnicodeEscapePartial
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Quick (30 mins)

9.623 Unicode Escape Syntax

Usage of the Unicode Escape syntax, with the `\u{xxxxx}` format, available since PHP 7.0.

```
<?php

// Produce an elephant icon in PHP 7.0+
echo \u{1F418};

// Produce the raw sequence in PHP 5.0
echo \u{1F418};

?>
```

See also PHP RFC: Unicode Codepoint Escape Syntax, Code point and Unicode.

Short name	Php/UnicodeEscapeSyntax
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and more recent
Severity	Major
Time To Fix	Slow (1 hour)

9.624 Uninitilized Property

Uninitilized properties are not fully bootstrapped at the end of the constructor.

Properties may be inited at definition time, along with their visibility and type. Some types are not inited at definition time, as any object, so they should be inited during constructor. At the end of the former, all properties shall have a legit value, and be ready for usage.

```
<?php

class x {
    private $foo = null;
    private $uninited;

    function __construct($arg) {
        $this->foo = $args;

        // $this->uninited is not inited, nor at definition, nor in constructor
        // it will hold null at the beginning of the next method call
    }
}

?>
```

9.624.1 Suggestions

- Remove the property, and move it to another class
- Add an initialisation for this property

Short name	Classes/UninitedProperty
Rulesets	*ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.625 Union Typehint

Union typehints allows the specification of several typehint for the same argument or return value. This is a PHP 8.0 new feature.

Several typehints are specified at the same place as a single one. The different values are separated by a pipe character `|`, like for exceptions

```php
<?php

// Example from the RFC https://wiki.php.net/rfc/union_types_v2
class Number {
    private int|float $number;

    public function setNumber(int|float $number): void {
        $this->number = $number;
    }

    public function getNumber(): int|float {
        return $this->number;
    }
}
?>
```

Union types are not compatible with PHP 7 and older.

See also PHP RFC: Union Types 2.0.

9.625.1 Suggestions

-

Short name	Php/Php80UnionTypehint
Rulesets	*CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*, *CompatibilityPHP74*
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)

9.626 Unitialized Properties

Properties that are not initialized in the constructor, nor at definition.

```php
<?php

class X {
    private $i1 = 1, $i2;
    protected $u1, $u2;

    function __construct() {
        $this->i2 = 1 + $this->u2;
    }

    function m() {
        echo $this->i1, $this->i2, $this->u1, $this->u2;
    }
}
?>
```

With the above class, when m() is accessed right after instantiation, there will be a missing property. Using default values at property definition, or setting default values in the constructor ensures that the created object is consistent.

9.626.1 Suggestions

- Add an explicit initialization for each property.

Short name	Classes/UnitializedProperties
Rulesets	*Suggestions*, *Top10*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*SPIP*

9.627 Unknown Parameter Name

The name of the parameter doesn't belong to the method signature.

```
<?php

// All good
foo(a:1, b:2, c:3);

// A is not a parameter name, it should be a
foo(A:1, b:2, c:3);

function foo($a, $b, $c) {}
?>
```

See also Named Arguments.

9.627.1 Suggestions

- Fix the name of the parameter and use a valid one
- Remove the parameter name, and revert to positional notation

Short name	Functions/UnknownParameterName
Rulesets	*Analyze*, *CI-checks*
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)

9.628 Unknown Pcre2 Option

`PCRE2` supports different options, compared to `PCRE1`. `PCRE2` was adopted with PHP 7.3.

The `S` modifier : it used to tell PCRE to spend more time studying the regex, so as to be faster at execution. This is now the default behavior, and may be dropped from the regex.

The X modifier : X is still existing with PCRE2, though it is now the default for PCRE2, and not for PHP as time of writing. In particular, Any backslash in a pattern that is followed by a letter that has no special meaning causes an error, thus reserving these combinations for future expansion. ``. It is recommended to avoid using useless sequence \s in regex to get ready for that change. All the following letters ``gijkmoqyFIJMOTY . Note that clLpPuU are valid PRCE sequences, and are probably failing for other reasons.

```
<?php

// \y has no meaning. With X option, this leads to a regex compilation error, and a␣
↪failed test.
preg_match('/ye\y/', $string);
preg_match('/ye\y/X', $string);

?>
```

See also Pattern Modifiers and PHP RFC: PCRE2 migration.

Short name	Php/UnknownPcre2Option
Rulesets	*Analyze*, *CompatibilityPHP73*
Severity	Minor
Time To Fix	Instant (5 mins)

9.629 Unkown Regex Options

Regex support in PHP accepts the following list of options : eimsuxADJSUX.

All other letter used as option are not supported : depending on the situation, they may be ignored or raise an error.

```
<?php

// all options are available
if (preg_match('/\d+/isA', $string, $results)) { }

// p and h are not regex options, p is double
if (preg_match('/\d+/php', $string, $results)) { }

?>
```

See also Pattern Modifiers

Short name	Structures/UnknownPregOption
Rulesets	*Analyze*
Severity	Major
Time To Fix	Slow (1 hour)

9.630 Unpacking Inside Arrays

The variadic operator is now available inside arrays. Until PHP 7.4, it is not possible to use the variadic operator, or ... inside arrays.

The workaround is to use array_merge(), after checking that arrays are not empty.

```
<?php

$a = ['a', 'b', 'c'];
$b = ['d', 'e', 'f'];

// PHP 7.4
$c = [...$a, ...$b];

// PHP 7.3 and older
$c = array_merge($a, $b);

?>
```

See also Spread Operator in Array Expression and PHP 5.6 and the Splat Operator .

9.630.1 Suggestions

- Replace array_merge() with `...`.

Short name	Php/UnpackingInsideArrays
Rule-sets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)

9.631 Unpreprocessed Values

Preprocessing values is the preparation of values before PHP executes the code.

There is no macro language in PHP, that prepares the code before compilation, bringing some comfort and short syntax. Most of the time, one uses PHP itself to preprocess data.

For example :

```
<?php
    $days_en = 'monday,tuesday,wednesday,thursday,friday,saturday,sunday';
    $days_zh = ',,,,,,';

    $days = explode(',', $lang === 'en' ? $days_en : $days_zh);
?>
```

could be written

```
<?php
    if ($lang === 'en') {
```

(continues on next page)

(continued from previous page)

```
        $days = ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday',
↪'sunday'];
    } else {
        $days = ['', '', '', '', '', '', ''];
    }
?>
```

and avoid preprocessing the string into an array first.

Preprocessing could be done anytime the script includes all the needed values to process the expression.

9.631.1 Suggestions

- Preprocess the values and hardcode them in PHP. Do not use PHP to calculate something at the last moment.
- Use already processed values, or cache to avoid calculating the value each hit.
- Create a class that export the data in the right format for every situation, including the developer's comfort.

Short name	Structures/Unpreprocessed
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Instant (5 mins)
ClearPHP	always-preprocess
Examples	*Zurmo*, *Piwigo*

9.632 Unreachable Class Constant

Class constants may be unreachable due to visibility configuration.

Since PHP 7.1, class constants support visibility. Their usage may be restricted to the current class, or `private`, to classes that extends or are extended by the current class, or `protected`. They may also be `public`, just like it was before.

```
<?php

class Foo{
    private const PRIVATE = 1;
            const PUBLIC = 3;
}

// PHP 7.1- and older
echo Foo::PUBLIC;

// This is not accessible
echo Foo::PRIVATE;

?>
```

See also Class Constant and PHP RFC: Support Class Constant Visibility.

9.632.1 Suggestions

- Make the class constant protected, when the call to the constant is inside a related class.
- Create another constant, that may be accessible
- Make the class constant public

Short name	Classes/UnreachableConstant
Rulesets	*ClassReview*
Severity	Major
Time To Fix	Slow (1 hour)

9.633 Unreachable Code

Code may be unreachable, because other instructions prevent its reaching.

For example, it be located after throw, return, exit() <https://www.php.net/‘exit>‘_, die() <https://www.php.net/‘die>‘_, goto, break or continue : this way, it cannot be reached, as the previous instruction will divert the engine to another part of the code.

```php
<?php

function foo() {
    $a++;
    return $a;
    $b++;      // $b++ can't be reached;
}

function bar() {
    if ($a) {
        return $a;
    } else {
        return $b;
    }
    $b++;      // $b++ can't be reached;
}

foreach($a as $b) {
    $c += $b;
    if ($c > 10) {
        continue 1;
    } else {
        $c--;
        continue;
    }
    $d += $e;   // this can't be reached
}

$a = 1;
goto B;
class foo {}    // Definitions are accessible, but not functioncalls
B:
echo $a;

?>
```

This is dead code, that may be removed.

9.633.1 Suggestions

- Remove the unreachable code
- Remove the blocking expression, and let the code execute

Short name	Structures/UnreachableCode
Rulesets	*Dead code*, *Suggestions*
Severity	Major
Time To Fix	Instant (5 mins)
ClearPHP	no-dead-code

9.634 Unresolved Catch

Catch clauses do not check for Exception existence.

Catch clauses check that the emitted expression is of the requested Class, but if that class doesn't exist in the code, the catch clause is always false. This is dead code.

```
<?php

try {
    // doSomething()
} catch {TypoedExxeption $e) { // Do not exist Exception
    // Fix this exception
} catch {Stdclass $e) {        // Exists, but is not an exception
    // Fix this exception
} catch {Exception $e) {        // Actual and effective catch
    // Fix this exception
}
?>
```

9.634.1 Suggestions

- Fix the name of the exception
- Remove the catch clause
- Add a use expression with a valid name
- Create/import the missing exception

Short name	Classes/UnresolvedCatch
Rulesets	*Dead code*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	no-unresolved-catch

9.635 Unresolved Classes

The following classes are instantiated in the code, but their definition couldn't be found.

```
<?php

class Foo extends Bar {
    private function foobar() {
        // here, parent is not resolved, as Bar is not defined in the code.
        return parent::$prop;
    }
}

?>
```

9.635.1 Suggestions

- Check for namespaces and aliases and make sure they are correctly configured.

Short name	Classes/UnresolvedClasses
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)

9.636 Unresolved Instanceof

The instanceof operator doesn't confirm if the compared class exists.

It checks if an variable is of a specific class. However, if the referenced class doesn't exist, because of a bug, a missed inclusion or a typo, the operator always fails, without a warning.

```
<?php

namespace X {
    class C {}

    // This is OK, as C is defined in X
    if ($o instanceof C) { }

    // This is not OK, as C is not defined in global
    // instanceof respects namespaces and use expressions
    if ($o instanceof \C) { }

    // This is not OK, as undefinedClass
    if ($o instanceof undefinedClass) { }

    // This is not OK, as $class is now a full namespace. It actually refers to \c,␣
↪which doesn't exist
    $class = 'C';
    if ($o instanceof $class) { }
}
?>
```

Make sure the following classes are well defined.

See also Instanceof.

9.636.1 Suggestions

- Remove the call to instanceof and all its dependencies.
- Fix the class name and use a class existing in the project.

Short name	Classes/UnresolvedInstanceof
Rulesets	*Analyze*, *Dead code*, *Top10*
Severity	Major
Time To Fix	Instant (5 mins)
ClearPHP	no-unresolved-instanceof
Examples	*WordPress*

9.637 Unresolved Use

The following use instructions cannot be resolved to a class or a namespace. They should be dropped or fixed.

```php
<?php

namespace A {
    // class B is defined
    class B {}
    // class C is not defined
}

namespace X/Y {

    use A/B;  // This use is valid
    use A/C;  // This use point to nothing.

    new B();
    new C();
}

?>
```

Use expression are options for the current namespace.

See also Using namespaces: Aliasing/Importing.

9.637.1 Suggestions

- Remove the use expression
- Fix the use expression

Short name	Namespaces/UnresolvedUse
Rulesets	*Analyze*
Severity	Major
Time To Fix	Slow (1 hour)
ClearPHP	no-unresolved-use

9.638 Unserialize Second Arg

Since PHP 7, unserialize() function has a second argument that limits the classes that may be unserialized. In case of a breach, this is limiting the classes accessible from unserialize().

One way to exploit unserialize, is to make PHP unserialized the data to an available class, may be one that may be auto-loaded.

```php
<?php

// safe unserialization : only the expected class will be extracted
$serialized = 'O:7:dbClass:0:{}';
$var = unserialize($serialized, ['dbClass']);
$var->connect();

// unsafe unserialization : $var may be of any type that was in the serialized string
// although, here, this is working well.
$serialized = 'O:7:dbClass:0:{}';
$var = unserialize($serialized);
$var->connect();

// unsafe unserialization : $var is not of the expected type.
// and, here, this will lead to disaster.
$serialized = 'O:10:debugClass:0:{}';
$var = unserialize($serialized);
$var->connect();

?>
```

See also unserialize(), Securely Implementing (De)Serialization in PHP, and Remote code execution via PHP [Unserialize].

9.638.1 Suggestions

- Add a list of class as second argument of any call to unserialize(). This is valid for PHP 7.0 and later.

Short name	Security/UnserializeSecondArg
Rulesets	*Security*
Php Version	With PHP 7.0 and more recent
Severity	Critical
Time To Fix	Quick (30 mins)
Examples	*Piwigo*, *LiveZilla*

9.639 Unset In Foreach

Unset applied to the variables of a `foreach` loop are useless. Those variables are copies and not the actual value. Even if the value is a reference, unsetting it has no effect on the original array : the only effect may be indirect, on elements inside an array, or on properties inside an object.

```
<?php

// When unset is useless
$array = [1, 2, 3];
foreach($array as $a) {
    unset($a);
}

print_r($array); // still [1, 2, 3]

foreach($array as $b => &$a) {
    unset($a);
}

print_r($array); // still [1, 2, 3]

// When unset is useful
$array = [ [ 'c' => 1] ]; // Array in array
foreach($array as &$a) {
    unset(&$a['c']);
}

print_r($array); // now [ ['c' => null] ]

?>
```

See also foreach.

9.639.1 Suggestions

- Drop the unset

Short name	Structures/UnsetInForeach
Rulesets	*Dead code*, *Analyze*
Severity	Major
Time To Fix	Quick (30 mins)

9.640 Unsupported Types With Operators

Arrays, resources and objects are generally not accepted with unary and binary operators.

The operators are +, -, *, /, **, %, <<, >>, &, |, ^, ~, ++ and –.

```
<?php

var_dump([] % [42]);
// int(0) in PHP 7.x
```

(continues on next page)

(continued from previous page)

```
// TypeError in PHP 8.0 +

// Also impossible usage : index are string or int
$a = [];
$b = $c[$a];

?>
```

In PHP 8.0, the rules have been made stricter and more consistent.

The only valid operator is +, combined with arrays in both operands. Other situation will throw *TypeError*.

See also Stricter type checks for arithmetic/bitwise operators and TypeError.

9.640.1 Suggestions

- Do not use those values with those operators
- Use a condition to skip this awkward situation
- Add an extra step to turn this value into a valid type

Short name	Structures/UnsupportedTypesWithOperators
Rulesets	*Analyze*, *CompatibilityPHP80*
Php Version	7.4-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Medium

9.641 Unthrown Exception

These are exceptions that are defined in the code but never thrown.

```
<?php

//This exception is defined but never used in the code.
class myUnusedException extends \Exception {}

//This exception is defined and used in the code.
class myUsedException extends \Exception {}

throw new myUsedException('I was called');

?>
```

See also Exceptions.

Short name	Exceptions/Unthrown
Rulesets	*Analyze*, *Dead code*
Severity	Minor
Time To Fix	Quick (30 mins)
ClearPHP	no-unthrown-exceptions

9.642 Unused Arguments

Those arguments are not used in the method or function.

Unused arguments should be removed in functions : they are just dead code.

Unused argument may have to stay in methods, as the signature is actually defined in the parent class.

```
<?php

// $unused is in the signature, but not used.
function foo($unused, $b, $c) {
    return $b + $c;
}
?>
```

9.642.1 Suggestions

- Drop the argument from the signature
- Actually use that argument in the body of the method

Short name	Functions/UnusedArguments
Rulesets	*Analyze*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*ThinkPHP*, *phpMyAdmin*

9.643 Unused Class Constant

The class constant is unused. Consider removing it.

```
<?php

class foo {
    public const UNUSED = 1; // No mention in the code

    private const USED = 2;  // used constant

    function bar() {
        echo self::USED;
    }
}

?>
```

9.643.1 Suggestions

- Remove the class constant
- Use the class constant

Short name	Classes/UnusedConstant
Rulesets	*Analyze*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.644 Unused Classes

The following classes are never explicitly used in the code.

Note that this may be valid in case the current code is a library or framework, since it defines classes that are used by other (unprovided) codes. Also, this analyzer may find classes that are, in fact, dynamically loaded.

```
<?php

class unusedClasss {}
class usedClass {}

$y = new usedClass();

?>
```

Short name	Classes/UnusedClass
Rulesets	*Dead code*, *Analyze*
Severity	Major
Time To Fix	Quick (30 mins)

9.645 Unused Constants

Those constants are defined in the code but never used. Defining unused constants slow down the application, as they are executed and stored in PHP hashtables.

```
<?php

// const-defined constant
const USED_CONSTANT  = 0;
const UNUSED_CONSTANT = 1 + USED_CONSTANT;

// define-defined constant
define('ANOTHER_UNUSED_CONSTANT', 3);

?>
```

It is recommended to comment them out, and only define them when it is necessary.

9.645.1 Suggestions

- Make use of the constant
- Remove the constant

Short name	Constants/UnusedConstants
Rulesets	*Dead code*
Severity	Minor
Time To Fix	Instant (5 mins)

9.646 Unused Functions

The functions below are unused. They look like dead code.

Recursive functions, level 1, are detected : they are only reported when a call from outside the function is made. Recursive functions calls of higher level (A calls B calls A) are not handled.

```
<?php

function used() {}
// The 'unused' function is defined but never called
function unused() {}

// The 'used' function is called at least once
used();

?>
```

9.646.1 Suggestions

- Use the function in the code
- Remove the functions from the code

Short name	Functions/UnusedFunctions
Rulesets	*Dead code*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Woocommerce*, *Piwigo*

9.647 Unused Global

A global keyword is used in a method, yet the variable is not actually used. This makes PHP import values for nothing, or may create interference

```
<?php
    function foo() {
        global bar;

        return 1;
    }
?>
```

9.647.1 Suggestions

- Remove the global declaration
- Remove the global variable altogether

Short name	Structures/UnusedGlobal
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Dolphin*

9.648 Unused Inherited Variable In Closure

Some closures forgot to make usage of inherited variables.

Closure have two separate set of incoming variables : the arguments (between parenthesis) and the inherited variables, in the 'use' clause. Inherited variables are extracted from the local environment at creation time, and keep their value until execution.

The reported closures are requesting some local variables, but do not make any usage of them. They may be considered as dead code.

```php
<?php

// In this closure, $y is forgotten, but $u is used.
$a = function ($y) use ($u) { return $u; };

// In this closure, $u is forgotten
$a = function ($y, $z) use ($u) { return $u; };

?>
```

See also Anonymous functions.

9.648.1 Suggestions

- Remove the unused inherited variable
- Make us of the unused inherited variable

Short name	Functions/UnusedInheritedVariable
Rulesets	*Analyze*, *Dead code*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*shopware*, *Mautic*

9.649 Unused Interfaces

Those interfaces are defined and never used. They should be removed, as they are dead code.

Interfaces may be use as parent for other interfaces, as typehint (argument, return and property), in instance of.

```
<?php

interface used {}
interface unused {}

// Used by implementation
class c implements used {}

// Used by extension
interface j implements used {}

$x = new c;

// Used in a instanceof
var_dump($x instanceof used);

// Used in a typehint
function foo(Used $x) {}

?>
```

9.649.1 Suggestions

- Remove the interface
- Actually use the interface

Short name	Interfaces/UnusedInterfaces
Rulesets	*Dead code*, *Suggestions*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Tine20*

9.650 Unused Label

Some labels have been defined in the code, but they are not used. They may be removed as they are dead code.

```
<?php

$a = 0;
A:

    ++$a;

    // A loop. A: is used
    if ($a < 10) { goto A; }

// B is never called explicitely. This is useless.
B:

?>
```

There is no analysis for undefined goto call, as PHP checks that goto has a destination label at compile time :

See also Goto.

9.650.1 Suggestions

- Remove the unused label
- Add a goto call to this label
- Check for spelling mistakes

Short name	Structures/UnusedLabel
Rulesets	*Dead code*
Severity	Minor
Time To Fix	Quick (30 mins)

9.651 Unused Methods

Those methods are never called.

They are probably dead code, unless they are called dynamically.

This analysis omits methods which are in a class that makes dynamical self calls : `$this->$m()`. That way, any method may be called.

This analysis omits methods which are overwritten by a child class. That way, they are considered to provide a default behavior.

```php
<?php

class foo {
    public function used() {
        $this->used();
    }

    public function unused() {
        $this->used();
    }
}

class bar extends foo {
    public function some() {
        $this->used();
    }
}

$a = new foo();
$a->used();

?>
```

See also Dead Code: Unused Method.

9.651.1 Suggestions

- Make use of the method

- Remove the method
- Move the method to another class

Short name	Classes/UnusedMethods
Rulesets	*Dead code*
Severity	Minor
Time To Fix	Slow (1 hour)

9.652 Unused Private Methods

Private methods that are not used are dead code.

Private methods are reserved for the defining class. Thus, they must be used with the current class, with `$this` or `self\:\:`.

Protected methods, in a standalone class, are also included.

```
<?php

class Foo {
    // Those methods are used
    private function method() {}
    private static function staticMethod() {}

    // Those methods are not used
    private function unusedMethod() {}
    private static function staticUnusedMethod() {}

    public function bar() {
        self::staticMethod();
        $this->method();
    }
}

?>
```

This analysis skips classes that makes self dynamic calls, such as `$this->$method()`.

9.652.1 Suggestions

- Remove the private method, as it is unused
- Add a call to this private method
- Change method visibility to make it available to other classes

Short name	Classes/UnusedPrivateMethod
Rulesets	*Dead code*
Severity	Minor
Time To Fix	Quick (30 mins)

9.653 Unused Private Properties

Unused static properties should be removed.

Unused private properties are dead code. They are usually leftovers of development or refactorisation : they used to have a mission, but are now left.

Being private, those properties are only accessible to the current class or trait. As such, validating the

```
<?php

class foo {
    // This is a used property (see bar method)
    private $used = 1;

    // This is an unused property
    private $unused = 2;

    function bar($a) {
        $this->used += $a;

        return $this->used;
    }
}

?>
```

9.653.1 Suggestions

- Remove the property altogether
- Check if the property wasn't forgotten in the rest of the class
- Check if the property is correctly named
- Change the visibility to protected or public : may be a visibility refactoring was too harsh

Short name	Classes/UnusedPrivateProperty
Rulesets	*Dead code*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*OpenEMR*, *phpadsnew*

9.654 Unused Protected Methods

The following protected methods are unused in children class. As such, they may be considered for being private.

Methods reported by this analysis are not used by children, yet they are protected.

```
<?php

class Foo {
    // This method is not used
    protected function unusedBar() {}
```

(continues on next page)

(continued from previous page)

```
    protected function usedInFoo() {}
    protected function usedInFooFoo() {}

    public function bar2() {
        // some code
        $this->usedInFoo();
    }
}

class FooFoo extends Foo {
    protected function bar() {}

    public function bar2() {
        // some code
        $this->usedInFooFoo();
    }
}

class someOtherClass {
    protected function bar() {
        // This is not related to foo.
        $this->unusedbar();
    }
}

?>
```

No usage of those methods were found.

This analysis is impacted by dynamic method calls.

9.654.1 Suggestions

- Make use of the protected method in the code
- Remove the method

Short name	Classes/UnusedProtectedMethods
Rulesets	*Dead code*
Severity	Major
Time To Fix	Slow (1 hour)

9.655 Unused Returned Value

The function called returns a value, which is ignored.

Usually, this is a sign of dead code, or a missed check on the results of the functioncall. At times, it may be a valid call if the function has voluntarily no return value.

It is recommended to add a check on the return value, or remove the call.

```
<?php
```

(continues on next page)

(continued from previous page)

```
// simplest form
function foo() {
    return 1;
}

foo();

// In case of multiple return, any one that returns something means that return value␣
↪is meaningful
function bar() {
    if (rand(0, 1)) {
        return 1;
    } else {
        return ;
    }
}

bar();

?>
```

Note that this analysis ignores functions that return void (same meaning that PHP 7.1 : return ; or no return in the function body).

Short name	Functions/UnusedReturnedValue
Rulesets	*Analyze*, *Dead code*
Severity	Minor
Time To Fix	Quick (30 mins)

9.656 Unused Trait In Class

A trait has been summoned in a class, but is not used. Traits may be used as a copy/paste of code, bringing a batch of methods and properties to a class. In the current case, the imported trait is never called. As such, it may be removed.

Currently, the analysis covers only traits that are used in the class where they are imported. Also, the properties are not covered yet.

```
<?php

trait t {
    function foo() { return 1;}
}

// this class imports and uses the trait
class UsingTrait {
    use t;

    function bar() {
        return $this->foo() + 1;
    }
}

// this class imports but doesn't uses the trait
```

(continues on next page)

(continued from previous page)

```
class UsingTrait {
    use t;

    function bar() {
        return 1;
    }
}

?>
```

There are some sneaky situations, where a trait falls into decay : for example, creating a method in the importing class, with the name of a trait class, will exclude the trait method, as the class method has priority. Other precedence rules may lead to the same effect.

See also Traits.

9.656.1 Suggestions

- Remove the trait from the class
- Actually use the trait, at least in the importing class
- Use conflict resolution to make the trait accessible

Short name	Traits/UnusedClassTrait
Rulesets	*ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)

9.657 Unused Use

Unused use statements. They may be removed, as they clutter the code and slows PHP by forcing it to search in this list for nothing.

```
<?php

use A as B; // Used in a new call.
use Unused; // Never used. May be removed

$a = new B();

?>
```

Short name	Namespaces/UnusedUse
Rulesets	*Dead code*
Severity	Major
Time To Fix	Instant (5 mins)
ClearPHP	no-useless-use

9.658 Unusual Case For PHP Functions

Usually, PHP functions are written all in lower case.

```
<?php

// All uppercases PHP functions
ECHO STRTOLOWER('This String');

?>
```

Short name	Php/UpperCaseFunction
Rulesets	*Coding Conventions*
Severity	Minor
Time To Fix	Instant (5 mins)

9.659 Upload Filename Injection

When receiving a file via Upload, it is recommended to store it under a self-generated name. Any storage that uses the original filename, or even a part of it may be vulnerable to injections.

```
<?php

// Security error ! the $_FILES['upload']['filename'] is provided by the sender.
// 'a.<script>alert(\'a\')</script>'; may lead to a HTML injection.
$extension = substr( strrchr($_FILES['upload']['name'], '.') ,1);
if (!in_array($extension, array('gif', 'jpeg', 'jpg')) {
    // process error
    continue;
}
// Md5 provides a name without special characters
$name = md5($_FILES['upload']['filename']);
if(@move_uploaded_file($_FILES['upload']['tmp_name'], '/var/no-www/upload/'.$name.'.'.
↪$extension)) {
    safeStoring($name.'.'.$extension, $_FILES['upload']['filename']);
}

// Security error ! the $_FILES['upload']['filename'] is provided by the sender.
if(@move_uploaded_file($_FILES['upload']['tmp_name'], $_FILES['upload']['filename']))
↪{
    safeStoring($_FILES['upload']['filename']);
}

// Security error ! the $_FILES['upload']['filename'] is provided by the sender.
// 'a.<script>alert('a')</script>'; may lead to a HTML injection.
$extension = substr( strrchr($_FILES['upload']['name'], '.') ,1);
$name = md5($_FILES['upload']['filename']);
if(@move_uploaded_file($_FILES['upload']['tmp_name'], $name.'.'.$extension)) {
    safeStoring($name.'.'.$extension, $_FILES['upload']['filename']);
}

?>
```

It is highly recommended to validate any incoming file, generate a name for it, and store the result in a folder outside the web folder. Also, avoid accepting PHP scripts, if possible.

See also [CVE-2017-6090], CWE-616: Incomplete Identification of Uploaded File Variables, Why File Upload Forms are a Major Security Threat.

9.659.1 Suggestions

- Validate uploaded filenames
- Rename files upon storage, and keep the original name in a database

Short name	Security/UploadFilenameInjection
Rulesets	*Security*
Severity	Major
Time To Fix	Instant (5 mins)

9.660 Use === null

It is faster to use === null instead of is_null().

```
<?php

// Operator === is fast
if ($a === null) {

}

// Function call is slow
if (is_null($a)) {

}


?>
```

9.660.1 Suggestions

- Use === comparison

Short name	Php/IsnullVsEqualNull
Rulesets	*Analyze*, *php-cs-fixable*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
ClearPHP	avoid-those-slow-functions

9.661 Use Array Functions

There are a lot of native PHP functions for arrays. It is often faster to take advantage of them than write a loop.

- array_push() : use array_merge()
- array_slice() : use array_chunk()
- index access : use array_column()
- append *[]*: use array_merge()
- addition : use array_sum()
- multiplication : use array_product()
- concatenation : use implode()
- ifthen : use array_filter()

```
<?php

$all = implode('-', $s).'-';

// same as above
$all = '';
foreach($array as $s) {
    $all .= $s . '-';
}

?>
```

See also Array Functions and *No array_merge() In Loops*.

9.661.1 Suggestions

- Remove the loop and use a native PHP function
- Add more expressions to the loop : batching multiple operations in one loop makes it more interesting than running separates loops.

Short name	Structures/UseArrayFunctions
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.662 Use Basename Suffix

basename() will remove extension when it is provided as argument. The second argument will be removed from the name of the file.

```
<?php

$path = 'phar:///path/to/file.php';

// Don't forget the .
$filename = basename($path, '.php');

// Too much work for this
$filename = substr(basename($path), 0, -4);
```

(continues on next page)

(continued from previous page)

```

?>
```

Using basename() instead of substr() or else, makes the intention clear.

See also basename.

9.662.1 Suggestions

- Use basename(), remove more complex code based on substr() or str_replace()

Short name	Structures/BasenameSuffix
Rulesets	*Suggestions*
Examples	*NextCloud*, *Dolibarr*

9.663 Use Case Value

When switch() has branched to the right case, the value of the switched variable is know : it is the case.

This doesn't work with complex expression cases, nor with default.

```
<?php

switch($a) {
    case 'a' :
        // $a == 'a';
        echo $a;
        break;

    case 'b' :
        // $a == 'b';
        echo 'b';
        break;
}

?>
```

9.663.1 Suggestions

- Use the literal value in the case, to avoid unnecessary computation.

Short name	Structures/UseCaseValue
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.664 Use Class Operator

Use `\:\:class` to hardcode class names, instead of strings.

This is actually faster than strings, which are parsed at execution time, while `\:\:class` is compiled, making it faster to execute.

It is also capable to handle aliases, making the code easier to maintain.

```
<?php

namespace foo\bar;

use foo\bar\X as B;

class X {}

$className = '\foo\bar\X';

$className = foo\bar\X::class;

$className = B\X;

$object = new $className;

?>
```

This is not possible when building the name of the class with concatenation.

This is a micro-optimization. This also helps static analysis, as it gives more information at compile time to analyse.

9.664.1 Suggestions

- Replace strings by the ::class operator whenever possible

Short name	Classes/UseClassOperator
Rulesets	*Analyze*, *Performances*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)

9.665 Use Const And Functions

Since PHP 5.6 it is possible to import specific functions or constants from other namespaces.

```
<?php

namespace A {
    const X = 1;
    function foo() { echo __FUNCTION__; }
}

namespace My{
    use function A\foo;
    use constant A\X;

    echo foo(X);
}
```

(continues on next page)

(continued from previous page)

```

?>
```

See also Using namespaces: Aliasing/Importing.

Short name	Namespaces/UseFunctionsConstants
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Php Version	With PHP 5.6 and more recent
Severity	Major
Time To Fix	Slow (1 hour)

9.666 Use Constant

The following functioncall have a constant equivalent, that is faster to use than calling the functions.

This applies to the following functions :

- pi() : replace with *M_PI*
- phpversion() : replace with *PHP_VERSION*
- php_sapi_name() : replace with *PHP_SAPI_NAME*

```
<?php

// recommended way
echo PHP_VERSION;

// slow version
echo php_version();

?>
```

See also PHP why 'pi() and M_PI <https://stackoverflow.com/questions/42021176/php-why-pi-and-m-pi>'_.

9.666.1 Suggestions

- Use the constant version, not the function.

Short name	Structures/UseConstant
Rulesets	*Analyze*, *php-cs-fixable*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)

9.667 Use Constant As Arguments

Some methods and functions are defined to be used with constants as arguments. Those constants are made to be meaningful and readable, keeping the code maintenable. It is recommended to use such constants as soon as they are documented.

```
<?php

// Turn off all error reporting
// 0 and -1 are accepted
error_reporting(0);

// Report simple running errors
error_reporting(E_ERROR | E_WARNING | E_PARSE);

// The first argument can be one of INPUT_GET, INPUT_POST, INPUT_COOKIE, INPUT_SERVER,
↪ or INPUT_ENV.
$search_html = filter_input(INPUT_GET, 'search', FILTER_SANITIZE_SPECIAL_CHARS);

// sort accepts one of SORT_REGULAR, SORT_NUMERIC, SORT_STRING, SORT_LOCALE_STRING,␣
↪SORT_NATURAL
// SORT_FLAG_CASE may be added, and combined with SORT_STRING or SORT_NATURAL
sort($fruits);

?>
```

Here is the list of function that use a unique PHP constant as argument :

- array_change_key_case()
- array_multisort()
- array_unique()
- count()
- dns_get_record()
- easter_days()
- extract()
- filter_input()
- filter_var()
- fseek()
- get_html_translation_table()
- gmp_div_q()
- gmp_div_qr()
- gmp_div_r()
- html_entity_decode()
- htmlspecialchars_decode()
- http_build_query()
- http_parse_cookie()
- http_parse_params()
- http_redirect()
- http_support()
- parse_ini_file()

- parse_ini_string()
- parse_url()
- pathinfo()
- pg_select()
- posix_access()
- round()
- scandir()
- socket_read()
- str_pad()
- trigger_error()

Here is the list of functions that use a combination of PHP native functions as argument.

- arsort()
- asort()
- error_reporting()
- filter_input()
- filter_var()
- get_html_translation_table()
- htmlentities()
- htmlspecialchars()
- http_build_url()
- jdtojewish()
- krsort()
- ksort()
- pg_result_status()
- phpcredits()
- phpinfo()
- preg_grep()
- preg_match()
- preg_split()
- rsort()
- runkit_import()
- sort()
- stream_socket_client()
- stream_socket_server()

9.667.1 Suggestions

- Use PHP native constants, whenever possible, instead of meaningless literals.

Short name	Functions/UseConstantAsArguments
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Tikiwiki*, *shopware*

9.668 Use Count Recursive

The code could use the recursive version of count.

The second argument of count, when set to COUNT_RECURSIVE, count recursively the elements. It also counts the elements themselves.

```
<?php

$array = array( array(1,2,3), array(4,5,6));

print (count($array, COUNT_RECURSIVE) - count($array, COUNT_NORMAL));

$count = 0;
foreach($array as $a) {
    $count += count($a);
}
print $count;

?>
```

See also count.

9.668.1 Suggestions

- Drop the loop and use the 2nd argument of count()

Short name	Structures/UseCountRecursive
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*WordPress*, *PrestaShop*

9.669 Use DateTimeImmutable Class

The DateTimeImmutable class is the immutable version of the Datetime class.

While DateTime may be modified 'in situ', DateTimeImmutable cannot be modified. Any modification to such an object will return a new and distinct object. This avoid interferences that are hard to track.

```
<?php
// Example extracted from Derick Rethans' article (link below)

function formatNextMondayFromNow( DateTime $dt )
{
        return $dt->modify( 'next monday' )->format( 'Y-m-d' );
}

$d = new DateTime();                            //2014-02-17
echo formatNextMondayFromNow( $d ), \n;
echo $d->format( 'Y-m-d' ), \n;               //2014-02-17
?>
```

See also What's all this 'immutable date' stuff, anyway?, DateTimeImmutable, The DateTime class and The DateTimeImmutable class.

9.669.1 Suggestions

- Always use DateTimeImmutable when manipulating dates.

Short name	Php/UseDateTimeImmutable
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.670 Use Instanceof

The `instanceof` operator is a more precise alternative to `is_object()`. It is also faster.

instanceof checks for an variable to be of a class or its parents or the interfaces it implements. Once `instanceof` has been used, the actual attributes available (properties, constants, methods) are known, unlike with `is_object()`.

Last, `instanceof` may be upgraded to Typehint, by moving it to the method signature.

```
<?php

class Foo {

    // Don't use is_object
    public function bar($o) {
        if (!is_object($o)) { return false; } // Classic argument check
        return $o->method();
    }

    // use instanceof
    public function bar($o) {
        if ($o instanceof myClass) {  // Now, we know which methods are available
            return $o->method();
        }

        return false; } // Default behavior
    }

```

(continues on next page)

(continued from previous page)

```
    // use of typehinting
    // in case $o is not of the right type, exception is raised automatically
    public function bar(myClass $o) {
        return $o->method();
    }
}

?>
```

`instanceof` and `is_object()` may not be always interchangeable. Consider using isset() on a known property for a simple check on objects. You may also consider is_string(), is_integer() or is_scalar(), in particular instead of `` !`is_object() <https://www.php.net/is_object>`_ ``.

The `instanceof` operator is also faster than the `is_object()` functioncall.

See also Type Operators and is_object.

9.670.1 Suggestions

- Use instanceof and remove is_object()
- Create a high level interface to check a whole family of classes, instead of testing them individually
- Use typehint when possible
- Avoid mixing scalar types and objects in the same variable

Short name	Classes/UseInstanceof
Rulesets	none
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*TeamPass*, *Zencart*

9.671 Use List With Foreach

Foreach() structures accepts list() as blind key. If the loop-value is an array with a fixed structure, it is possible to extract the values directly into variables with explicit names.

```
<?php

// Short way to assign variables
// Works on PHP 7.1, where list() accepts keys.
foreach($names as list('first' => $first, 'last' => $last)) {
    doSomething($first, $last);
}

// Short way to assign variables
// Works on all PHP versions with numerically indexed arrays.
foreach($names as list($first, $last)) {
    doSomething($first, $last);
}

// Long way to assign variables
```

(continues on next page)

(continued from previous page)

```
foreach($names as $name) {
    $first = $name['first'];
    $last = $name['last'];

    doSomething($first, $last);
}

?>
```

See also list and foreach.

9.671.1 Suggestions

- Use the list keyword (or the short syntax), and simplify the array calls in the loop.

Short name	Structures/UseListWithForeach
Rulesets	*Suggestions*, *Top10*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*MediaWiki*

9.672 Use Lower Case For Parent, Static And Self

The special parent, static and self keywords needed to be lowercase to be usable. This was fixed in PHP 5.5; otherwise, they would yield a 'PHP Fatal error: Class 'PARENT' not found'.

parent, static and self are traditionally written in lowercase only. Mixed case and Upper case are both valid, though.

```
<?php

class foo {
    const aConstante = 233;

    function method() {
        // Wrong case, error with PHP 5.4.* and older
        echo SELF::aConstante;

        // Always right.
        echo self::aConstante;
    }
}

?>
```

Until PHP 5.5, non-lowercase version of those keywords are generating a bug.

Short name	Php/CaseForPSS
Rulesets	*CompatibilityPHP54*, *CompatibilityPHP53*
Php Version	With PHP 5.5 and older
Severity	Minor
Time To Fix	Instant (5 mins)

9.673 Use Named Boolean In Argument Definition

Boolean in argument definitions is confusing.

It is recommended to use explicit constant names, instead. They are more readable. They also allow for easy replacement when the code evolve and has to replace those booleans by strings. This works even also with classes, and class constants.

```
<?php

function flipImage($im, $horizontal = NO_HORIZONTAL_FLIP, $vertical = NO_VERTICAL_
↪FLIP) { }

// with constants
const HORIZONTAL_FLIP = true;
const NO_HORIZONTAL_FLIP = true;
const VERTICAL_FLIP = true;
const NO_VERTICAL_FLIP = true;

rotateImage($im, HORIZONTAL_FLIP, NO_VERTICAL_FLIP);


// without constants
function flipImage($im, $horizontal = false, $vertical = false) { }

rotateImage($im, true, false);

?>
```

See also Flag Argument, to avoid boolean altogether.

Short name	Functions/AvoidBooleanArgument
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*phpMyAdmin*, *Cleverstyle*

9.674 Use Nullable Type

The code uses nullable type, available since PHP 7.1.

Nullable Types are an option to type hint : they allow the passing value to be null, or another type.

According to the authors of the feature : ‘It is common in many programming languages including PHP to allow a variable to be of some type or null. This null often indicates an error or lack of something to return.’

```
<?php

function foo(?string $a = 'abc') : ?string {
    return $a.b;
}

?>
```

See also Type declarations and PHP RFC: Nullable Types.

Short name	Php/UseNullableType
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP70*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.1 and more recent
Severity	Major
Time To Fix	Quick (30 mins)

9.675 Use PHP Object API

OOP API is the modern version of the PHP API.

When PHP offers the alternative between procedural and OOP api for the same features, it is recommended to use the OOP API.

Often, this least to more compact code, as methods are shorter, and there is no need to bring the resource around. Lots of new extensions are directly written in OOP form too.

OOP / procedural alternatives are available for mysqli, tidy, cairo, finfo, and some others.

```
<?php
/// OOP version
$mysqli = new mysqli(localhost, my_user, my_password, world);

/* check connection */
if ($mysqli->connect_errno) {
    printf(Connect failed: %s\n, $mysqli->connect_error);
    exit();
}

/* Create table doesn't return a resultset */
if ($mysqli->query(CREATE TEMPORARY TABLE myCity LIKE City) === TRUE) {
    printf(Table myCity successfully created.\n);
}

/* Select queries return a resultset */
if ($result = $mysqli->query(SELECT Name FROM City LIMIT 10)) {
    printf(Select returned %d rows.\n, $result->num_rows);

    /* free result set */
    $result->close();
}
?>
```

```
<?php
/// Procedural version
$link = mysqli_connect(localhost, my_user, my_password, world);

/* check connection */
if (mysqli_connect_errno()) {
    printf(Connect failed: %s\n, mysqli_connect_error());
    exit();
```

(continues on next page)

(continued from previous page)

```
}

/* Create table doesn't return a resultset */
if (mysqli_query($link, CREATE TEMPORARY TABLE myCity LIKE City) === TRUE) {
    printf(Table myCity successfully created.\n);
}

?>
```

9.675.1 Suggestions

- Use the object API

Short name	Php/UseObjectApi
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Slow (1 hour)
ClearPHP	use-object-api
Examples	*WordPress*, *PrestaShop*, *SugarCrm*

9.676 Use PHP7 Encapsed Strings

PHP 7 has optimized the handling of double-quoted strings. In particular, double-quoted strings are much less memory hungry than classic concatenations.

PHP allocates memory at the end of the double-quoted string, making only one call to the allocator. On the other hand, concatenations are allocated each time they include dynamic content, leading to higher memory consumption.

```
<?php

$bar = 'bar';

/* PHP 7 optimized this */
$a = "foo and $bar";

/* This is PHP 5 code (aka, don't use it) */
$a = 'foo and ' . $bar;

// Constants can't be used with double quotes
$a = 'foo and ' . __DIR__;
$a = foo and __DIR__; // __DIR__ is not interpolated

?>
```

Concatenations are still needed with constants, static constants, magic constants, functions, static properties or static methods.

See also PHP 7 performance improvements (3/5): Encapsed strings optimization.

Short name	Performances/PHP7EncapsedStrings
Rulesets	*Performances*

9.677 Use Pathinfo

Use pathinfo() function instead of string manipulations.

pathinfo() is more efficient and readable and string functions.

```
<?php

$filename = '/path/to/file.php';

// With pathinfo();
$details = pathinfo($filename);
print $details['extension'];  // also capture php

// With string functions (other solutions possible)
$ext = substr($filename, - strpos(strreverse($filename), '.')); // Capture php

?>
```

When the path contains UTF-8 characters, pathinfo() may strip them. There, string functions are necessary.

9.677.1 Suggestions

- Use pathinfo() and its second argument

Short name	Php/UsePathinfo
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*SuiteCrm*

9.678 Use Positive Condition

Whenever possible, use a positive condition.

Positive conditions are easier to understand, and lead to less understanding problems. Negative conditions are not reported when else is not present.

```
<?php

// This is a positive condition
if ($a == 'b') {
    doSomething();
} else {
    doSomethingElse();
}

if (!empty($a)) {
    doSomething();
} else {
    doSomethingElse();
}

```

(continues on next page)

(continued from previous page)

```
// This is a negative condition
if ($a == 'b') {
    doSomethingElse();
} else {
    doSomething();
}

// No need to force $a == 'b' with empty else
if ($a != 'b') {
    doSomethingElse();
}


?>
```

9.678.1 Suggestions

- Invert the code in the if branches, and the condition

Short name	Structures/UsePositiveCondition
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*SPIP*, *ExpressionEngine*

9.679 Use System Tmp

It is recommended to avoid hardcoding the temporary file. It is better to rely on the system's temporary folder, which is accessible with sys_get_temp_dir().

```
<?php

// Where the tmp is :
file_put_contents(sys_get_temp_dir().'/tempFile.txt', $content);


// Avoid hard-coding tmp folder :
// On Linux-like systems
file_put_contents('/tmp/tempFile.txt', $content);

// On Windows systems
file_put_contents('C:\WINDOWS\TEMP\tempFile.txt', $content);

?>
```

See also PHP: When is /tmp not /tmp?.

9.679.1 Suggestions

- Do not hardcode the temporary file, use the system's

Short name	Structures/UseSystemTmp
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Slow (1 hour)

9.680 Use The Blind Var

When in a loop, it is faster to rely on the blind var, rather than the original source.

When the key is referenced in the foreach loop, it is faster to use the available container to access a value for reading.

Note that it is also faster to use the value with a reference to handle the writings.

```
<?php

// Reaching $source[$key] via $value is faster
foreach($source as $key => $value) {
    $coordinates = array('x' => $value[0],
                         'y' => $value[1]);
}

// Reaching $source[$key] via $source is slow
foreach($source as $key => $value) {
    $coordinates = array('x' => $source[$key][0],
                         'y' => $source[$key][1]);
}

?>
```

9.680.1 Suggestions

- Use the blind var

Short name	Performances/UseBlindVar
Rulesets	*Performances*
Severity	Minor
Time To Fix	Instant (5 mins)

9.681 Use Url Query Functions

PHP features several functions dedicated to processing URL's query string.

- parse_str()
- parse_url()
- http_build_query()

Those functions include extra checks : for example, http_build_query() adds urlencode() call on the values, and allow for choosing the separator and the Query string format.

```
<?php
$data = array(
    'foo' => 'bar',
    'baz' => 'boom',
    'cow' => 'milk',
    'php' => 'hypertext processor'
);

// safe and efficient way to build a query string
echo http_build_query($data, '', '&') . PHP_EOL;

// slow way to produce a query string
foreach($data as $name => &$value) {
    $value = $name.'='.$value;
}
echo implode('&', $data) . PHP_EOL;

?>
```

9.681.1 Suggestions

-

Short name	Structures/UseUrlQueryFunctions
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.682 Use With Fully Qualified Name

Use statement doesn't require a fully qualified name.

PHP manual recommends not to use fully qualified name (starting with) when using the 'use' statement : they are "the leading backslash is unnecessary and not recommended, as import names must be fully qualified, and are not processed relative to the current namespace".

```
<?php

// Recommended way to write a use statement.
use  A\B\C\D as E;

// No need to use the initial \
use \A\B\C\D as F;

?>
```

9.682.1 Suggestions

- Remove the initial in use expressions.

Short name	Namespaces/UseWithFullyQualifiedNS
Rulesets	*Analyze*, *Coding Conventions*
Severity	Minor
Time To Fix	Slow (1 hour)

9.683 Use array_slice()

Array_slice is de equivalent of substr() for arrays.

array_splice() is also possible, to remove a portion of array inside the array, not at the ends. This has no equivalent for strings.

```
<?php

$array = range(0, 9);

// Extract the 5 first elements
print_r(array_slice($array, 0, 5));

// Extract the 4 last elements
print_r(array_slice($array, -4));

// Extract the 2 central elements : 4 and 5
print_r(array_splice($array, 4, 2));

// slow way to remove the last elementst of an array
for($i = 0; $i < 4) {
    array_pop($array);
}

?>
```

See also array_slice and array_splice.

9.683.1 Suggestions

-

Short name	Performances/UseArraySlice
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.684 Use const

The const keyword may be used to define constant, just like the define() function.

When defining a constant, it is recommended to use 'const' when the features of the constant are not dynamical (name or value are known at compile time). This way, constant will be defined at compile time, and not at execution time.

```
<?php
  //Do
  const A = 1;
  // Don't
  define('A', 1);

?>
```

define() function is useful when the constant is not known at compile time, or when case sensitivity is necessary.

```
<?php
  // Read $a in database or config file
  define('A', $a);

  // Read $a in database or config file
  define('B', 1, true);
  echo b;
?>
```

See also Syntax.

9.684.1 Suggestions

- Use const instead of define()

Short name	Constants/ConstRecommended
Rulesets	*Analyze*, *Coding Conventions*, *Top10*, *CI-checks*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*phpMyAdmin*, *Piwigo*

9.685 Use is_countable

is_countable() checks if a variables holds a value that can be counted. It is recommended to use it before calling count().

is_countable() accepts arrays and object whose class implements countable.

```
<?php

function foo($arg) {
    if (!is_countable($arg)) {
        // $arg cannot be passed to count()
        return 0
    }
    return count($arg);
}

function bar($arg) {
    if (!is_array($arg) and !$x instanceof \Countable) {
        // $arg cannot be passed to count()
        return 0
    }
```

(continues on next page)

(continued from previous page)

```
    return count($arg);
}

?>
```

See also PHP RFC: is_countable.

9.685.1 Suggestions

- Use is_countable()
- Create a compatibility function that replaces is_countable() until the code is ready for PHP 7.3

Short name	Php/CouldUseIsCountable
Rulesets	*Suggestions*
Php Version	With PHP 7.3 and more recent

9.686 Use json_decode() Options

json_decode() returns objects by default, unless the second argument is set to `TRUE` or `JSON_OBJECT_AS_ARRAY`. Then, it returns arrays.

Avoid casting the returned value from json_decode(), and use the second argument to directly set the correct type.

```
<?php

$json = '{a:b}';

// Good syntax
$array = json_decode($json, JSON_OBJECT_AS_ARRAY);

// GoToo much work
$array = (array) json_decode($json);

?>
```

Note that all objects will be turned into arrays, recursively. If you're expecting an array of objects, don't use the `JSON_OBJECT_AS_ARRAY` constant, and change your JSON code.

Note that `JSON_OBJECT_AS_ARRAY` is the only constant : there is no defined constant to explicitly ask for an object as returned value.

See also json_decode.

9.686.1 Suggestions

- Use the correct second argument of json_decode() : JSON_OBJECT_AS_ARRAY

Short name	Structures/JsonWithOption
Rulesets	*Suggestions*
Severity	Major
Time To Fix	Quick (30 mins)

9.687 Use password_hash()

password_hash() and password_check() are a better choice to replace the use of crypt() to check password.

PHP 5.5 introduced these functions.

```
<?php

$password = 'rasmuslerdorf';
$hash = '$2y$10$YCFsG6elYca568hBi2pZ0.3LDL5wjgxct1N8w/oLR/jfHsiQwCqTS';

// The cost parameter can change over time as hardware improves
$options = array('cost' => 11);

// Verify stored hash against plain-text password
if (password_verify($password, $hash)) {
    // Check if a newer hashing algorithm is available
    // or the cost has changed
    if (password_needs_rehash($hash, PASSWORD_DEFAULT, $options)) {
        // If so, create a new hash, and replace the old one
        $newHash = password_hash($password, PASSWORD_DEFAULT, $options);
    }

    // Log user in
}
?>
```

See also Password hashing.

Short name	Php/Password55
Rulesets	*CompatibilityPHP55*
Php Version	With PHP 5.5 and more recent
Severity	Major
Time To Fix	Slow (1 hour)

9.688 Use pathinfo() Arguments

pathinfo() has a second argument to select only useful data.

It is twice faster to get only one element from pathinfo() than get the four of them, and use only one.

This analysis reports pathinfo() usage, without second argument, where only one or two indices are used, after the call.

```
<?php

// This could use only PATHINFO_BASENAME
function foo_db() {
```

(continues on next page)

(continued from previous page)

```
    $a = pathinfo($file2);
    return $a['basename'];
}

// This could be 2 calls, with PATHINFO_BASENAME and PATHINFO_DIRNAME.
function foo_de() {
    $a = pathinfo($file3);
    return $a['dirname'].'/'.$a['basename'];
}

// This is OK : 3 calls to pathinfo() is slower than array access.
function foo_deb() {
    $a = pathinfo($file4);
    return  $a['dirname'].'/'.$a['filename'].'.'.$a['extension'];
}

?>
```

Depending on the situation, the functions dirname() and basename() may also be used. They are even faster, when only fetching those data.

See also list.

9.688.1 Suggestions

- Use PHP native function pathinfo() and its arguments

Short name	Php/UsePathinfoArgs
Rulesets	*Performances*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Zend-Config*, *ThinkPHP*

9.689 Use random_int()

rand() and mt_rand() should be replaced with random_int().

At worse, rand() should be replaced with mt_rand(), which is a drop-in replacement and srand() by mt_srand().

random_int() replaces rand(), and has no seeding function like srand().

Other sources of entropy that should be replaced by random_int() : microtime(), uniqid(), time(). Those a often combined with hashing functions and mixed with other sources of entropy, such as a salt.

```
<?php

// Avoid using this
$random = rand(0, 10);

// Drop-in replacement
$random = mt_rand(0, 10);

// Even better but different :
```

(continues on next page)

(continued from previous page)

```
// valid with PHP 7.0+
try {
    $random = random_int(0, 10);
} catch (\Exception $e) {
    // process case of not enoug random values
}

// This is also a source of entropy, based on srand()
// random_int() is a drop-in replacement here
$a = sha256(uniqid());

?>
```

Since PHP 7, random_int() along with random_bytes(), provides cryptographically secure pseudo-random bytes, which are good to be used when security is involved. openssl_random_pseudo_bytes() may be used when the `OpenSSL` extension is available.

See also CSPRNG and OpenSSL.

9.689.1 Suggestions

- Use random_bytes() and randon_int(). At least, use them as a base for random data, and then add extra prefix and suffix, and a hash call on top.

Short name	Php/BetterRand
Rulesets	*Analyze*, *Security*, *CompatibilityPHP71*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)
Examples	*Thelia*, *FuelCMS*

9.690 Use session_start() Options

It is possible to set the session's option at session_start() call, skipping the usage of session_option().

This way, session's options are set in one call, saving several hits.

This is available since PHP 7.0. It is recommended to set those values in the `php.ini` file, whenever possible.

```
<?php

// PHP 7.0
session_start(['session.name' => 'mySession',
               'session.cookie_httponly' => 1,
               'session.gc_maxlifetime' => 60 * 60);

// PHP 5.6- old way
ini_set ('session.name', 'mySession');
ini_set(session.cookie_httponly, 1);
ini_set('session.gc_maxlifetime', 60 * 60);
session_start();

?>
```

9.690.1 Suggestions

- Use session_start() with array arguments

Short name	Php/UseSessionStartOptions
Rulesets	*Suggestions*
Php Version	With PHP 7.0 and more recent
Examples	*WordPress*

9.691 Used Once Property

Property used once in their defining class.

Properties used in one method only may be used several times, and read only. This may be a class constant. Such properties are meant to be overwritten by an extending class, and that's possible with class constants.

Setting properties with default values is a good way to avoid littering the code with literal values, and provide a single point of update (by extension, or by hardcoding) for all those situations. A constant is definitely better suited for this task.

```php
<?php

class foo {
    private $defaultCols = '*';
    cont DEFAULT_COLUMNS = '*';

    // $this->defaultCols holds a default value. Should be a constant.
    function bar($table, $cols) {
        // This is necessary to activate usage of default values
        if (empty($cols)) {
            $cols = $this->defaultCols;
        }
        $res = $this->query('SELECT '.$cols.' FROM '.$table);
        // ....
    }

    // Upgraded version of bar, with default values
    function bar2($table, $cols = self::DEFAULT_COLUMNS) {
        $res = $this->query('SELECT '.$cols.' FROM '.$table);
        // .....
    }
}

?>
```

9.691.1 Suggestions

- Remove the property, as it is probably not unused
- Add another usage of the property where it is useful

Short name	Classes/UsedOnceProperty
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

9.692 Used Once Variables

This is the list of used once variables.

```php
<?php

// The variables below never appear again in the code
$writtenOnce = 1;

foo($readOnce);

?>
```

Such variables are useless. Variables must be used at least twice : once for writing, once for reading, at least. It is recommended to remove them.

In special situations, variables may be used once :

- PHP predefined variables, as they are already initialized. They are omitted in this analyze.
- Interface function's arguments, since the function has no body; They are omitted in this analyze.
- Dynamically created variables ($$x, ${$this->y} or also using extract), as they are runtime values and can't be determined at static code time. They are reported for manual review.
- Dynamically included files will provide in-scope extra variables.

The current analyzer count variables at the application level, and not at a method scope level.

9.692.1 Suggestions

- Remove the variable
- Fix the name of variable
- Use the variable a second time, at least

Short name	Variables/VariableUsedOnce
Rulesets	*Analyze*, *Top10*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*shopware*, *Vanilla*

9.693 Used Once Variables (In Scope)

This is the list of used once variables, scope by scope. Those variables are used once in a function, a method, a class or a namespace. In any case, this means the variable is read or written, while it should be used at least twice.

```
<?php

function foo() {
    // The variables below never appear twice, inside foo()
    $writtenOnce = 1;

    foo($readOnce);
    // They do appear again in other functions, or in global space.
}

function bar() {
    $writtenOnce = 1;
    foo($readOnce);
}

?>
```

9.693.1 Suggestions

- Remove the variable
- Fix the name of variable
- Use the variable a second time in the current scope, at least

Short name	Variables/VariableUsedOnceByContext
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*shopware*

9.694 Useless Abstract Class

Those classes are marked 'abstract' and they are never extended. This way, they won't be instantiated nor used.

Abstract classes that have only static methods are omitted here : one usage of such classes are Utilities classes, which only offer static methods.

```
<?php

// Never extended class : this is useless
abstract class foo {}

// Extended class
abstract class bar {
    public function barbar() {}
}

class bar2 extends bar {}

// Utility class : omitted here
abstract class bar {
```

(continues on next page)

(continued from previous page)

```
    public static function barbar() {}
}

?>
```

9.694.1 Suggestions

- Drop the abstract keyword
- Actually add an abstract keyword

Short name	Classes/UselessAbstract
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)

9.695 Useless Alias

It is not possible to declare an alias of a method with the same name.

PHP reports that `Trait method f has not been applied, because there are collisions with other trait methods on x`, which is a way to say that the alias will be in conflict with the method name.

When the method is the only one bearing a name, and being imported, there is no need to alias it. When the method is imported in several traits, the keyword `insteadof` is available to solve the conflict.

```
<?php

trait t {
    function h() {}
}

class x {
    use t {
        // This is possible
        t::f as g;

        // This is not possible, as the alias is in conflict with itself
        // alias are case insensitive
        t::f as f;
    }
}

?>
```

This code lints but doesn't execute.

See also Conflict resolution.

9.695.1 Suggestions

- Remove the alias
- Fix the alias or the origin method name
- Switch to insteadof, and avoid as keyword

Short name	Traits/UselessAlias
Rulesets	*Analyze*, *LintButWontExec*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)

9.696 Useless Brackets

Standalone brackets have no use. Brackets are used to delimit a block of code, and are used by control statements. They may also be used to protect variables in strings.

Standalone brackets may be a left over of an old instruction, or a misunderstanding of the alternative syntax.

```
<?php

// The following brackets are useless : they are a leftover from an older instruction
// if (DEBUG)
{
    $a = 1;
}

// Here, the extra brackets are useless
for($a = 2; $a < 5; $a++) : {
    $b++;
} endfor;

?>
```

9.696.1 Suggestions

- Remove the brackets
- Restore the flow-control operation that was there and removed
- Move the block into a method or function, and call it

Short name	Structures/UselessBrackets
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*ChurchCRM*, *Piwigo*

9.697 Useless Casting

There is no need to overcast returned values.

```
<?php

// trim always returns a string : cast is useless
$a = (string) trim($b);

// strpos doesn't always returns an integer : cast is useful
$a = (boolean) strpos($b, $c);

// comparison don't need casting, nor parenthesis
$c = (bool) ($b > 2);

?>
```

See also Type juggling.

9.697.1 Suggestions

- Remove the type cast

Short name	Structures/UselessCasting
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*FuelCMS*, *ThinkPHP*

9.698 Useless Catch

Catch clause should handle the exception with some work.

Among the task of a catch clause : log the exception, clean any mess that was introduced, fail graciously.

```
<?php

function foo($a) {
    try {
        $b = doSomething($a);
    } catch (Throwable $e) {
        // No log of the exception : no one knows it happened.

        // return immediately ?
        return false;
    }

    $b->complete();

    return $b;
}

?>
```

See also Exceptions and Best practices for PHP exception handling.

9.698.1 Suggestions

- Add a log call to the catch block
- Handle correctly the exception

Short name	Exceptions/UselessCatch
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*Zurmo*, *PrestaShop*

9.699 Useless Check

There is no need to check the size of an array content before using foreach. Foreach() applies a test on the source, and skips the loop if no element is found.

```php
<?php

// Checking for type is good.
if (is_array($array)) {
    foreach($array as $a) {
        doSomething($a);
    }
}

// Foreach on empty arrays doesn't start. Checking is useless
if (!empty($array)) {
    foreach($array as $a) {
        doSomething($a);
    }
}

?>
```

This analysis checks for conditions with sizeof() and count(). Conditions with isset() and empty() are omitted : they also check for the variable existence, and thus, offer extra coverage.

See also foreach.

9.699.1 Suggestions

- Drop the condition and the check
- Turn the condition into isset(), empty() and is_array()

Short name	Structures/UselessCheck
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Magento*, *Phinx*

9.700 Useless Constructor

Class constructor that have empty bodies are useless. They may be removed.

```php
<?php

class X {
    function __construct() {
        // Do nothing
    }
}

class Y extends X {
    // Useful constructor, as it prevents usage of the parent
    function __construct() {
        // Do nothing
    }
}

?>
```

Short name	Classes/UselessConstructor
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)

9.701 Useless Default Argument

One of the argument has a default value, and this default value is never used. Every time the method is called, the argument is provided explicitly, rendering the default value actually useless.

```php
<?php

function goo($a, $b = 3) {
    // do something here
}

// foo is called 3 times, and sometimes, $b is not provided.
goo(1,2);
goo(1,2);
goo(1);


function foo($a, $b = 3) {
    // do something here
}

// foo is called 3 times, and $b is always provided.
foo(1,2);
foo(1,2);
foo(1,2);
?>
```

9.701.1 Suggestions

- Remove the default value
- Remove the explicit argument in the function call, when it is equal to the default value

Short name	Functions/UselessDefault
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)

9.702 Useless Final

When a class is declared final, all of its methods are also final by default.

There is no need to declare them individually final.

```php
<?php

    final class foo {
        // Useless final, as the whole class is final
        final function method() { }
    }

    class bar {
        // Useful final, as the whole class is not final
        final function method() { }
    }

?>
```

See also Final Keyword, and When to declare final.

Short name	Classes/UselessFinal
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
ClearPHP	no-useless-final

9.703 Useless Global

Global are useless in two cases. First, on super-globals, which are always globals, like $_GET; secondly, on variables that are not used.

```php
<?php

// $_POST is already a global : it is in fact a global everywhere
global $_POST;

// $unused is useless
function foo() {
    global $used, $unused;
```

(continues on next page)

(continued from previous page)

```
    ++$used;
}

?>
```

Also, PHP has superglobals, a special team of variables that are always available, whatever the context. They are : $GLOBALS, $_SERVER, $_GET, $_POST, $_FILES, $_COOKIE, $_SESSION, $_REQUEST and $_ENV.

9.703.1 Suggestions

- Drop the global expression

Short name	Structures/UselessGlobal
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*Zencart*, *HuMo-Gen*

9.704 Useless Instructions

Those instructions are useless, or contains useless parts.

For example, an addition whose result is not stored in a variable, or immediately used, does nothing : it is actually performed, and the result is lost. Just plain lost. In fact, PHP might detect it, and optimize it away.

Here the useless instructions that are spotted :

```
<?php

// Concatenating with an empty string is useless.
$string = 'This part '.$is.' useful but '.$not.'';

// This is a typo, that PHP turns into a constant, then a string, then nothing.
continue;

// Empty string in a concatenation
$a = 'abc' . '';

// Returning expression, whose result is not used (additions, comparisons, properties,
↪ closures, new without =, ...)
1 + 2;

// Returning post-incrementation
function foo($a) {
    return $a++;
}

// array_replace() with only one argument
$replaced = array_replace($array);
// array_replace() is OK with ...
$replaced = array_replace(...$array);
```

(continues on next page)

(continued from previous page)

```

// @ operator on source array, in foreach, or when assigning literals
$array = @array(1,2,3);

// Multiple comparisons in a for loop : only the last is actually used.
for($i = 0; $j = 0; $j < 10, $i < 20; ++$j, ++$i) {
    print $i.' '.$j.PHP_EOL;
}

// Counting the keys and counting the array is the same.
$c = count(array_keys($array))

//array_keys already provides an array with only unique values, as they were keys in␣
↪a previous array
$d = array_unique(array_keys($file['messages']))

// No need for assignation inside the ternary operator
$closeQuote = $openQuote[3] === "'" ? substr($openQuote, 4, -2) : $closeQuote =␣
↪substr($openQuote, 3);

?>
```

9.704.1 Suggestions

- Remove the extra semi-colon
- Remove the useless instruction
- Assign this expression to a variable and make use of it

Short name	Structures/UselessInstruction
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-useless-instruction

9.705 Useless Interfaces

The interfaces below are defined and are implemented by some classes.

However, they are never used to enforce an object's class in the code, using instanceof or in a typehint. As they are currently used, those interfaces may be removed without change in behavior.

```
<?php
    // only defined interface but never enforced
    interface i {};
    class c implements i {}
?>
```

Interfaces should be used in Typehint or with the instanceof operator.

```
<?php
    interface i {};

    function foo(i $arg) {
        // Now, $arg is always an 'i'
    }

    function bar($arg) {
        if (!($arg instanceof i)) {
            // Now, $arg is always an 'i'
        }
    }
?>
```

9.705.1 Suggestions

- Use the interface with instanceof, or a typehint
- Drop the interface altogether : both definition and implements keyword

Short name	Interfaces/UselessInterfaces
Rulesets	*Analyze*, *ClassReview*, *Typechecks*
Severity	Minor
Time To Fix	Quick (30 mins)
ClearPHP	no-useless-interfaces
Examples	*Woocommerce*

9.706 Useless Parenthesis

Situations where parenthesis are not necessary, and may be removed.

Parenthesis group several elements together, and allows for a more readable expression. They are used with logical and mathematical expressions. They are necessary when the precedence of the operators are not the intended execution order : for example, when an addition must be performed before the multiplication.

Sometimes, the parenthesis provide the same execution order than the default order : they are deemed useless.

```
<?php

    if ( ($condition) ) {}
    while( ($condition) ) {}
    do $a++; while ( ($condition) );

    switch ( ($a) ) {}
    $y = (1);
    ($y) == (1);

    f(($x));

    // = has precedence over ==
    ($a = $b) == $c;

    ($a++);
```

(continues on next page)

(continued from previous page)

```

    // No need for parenthesis in default values
    function foo($c = ( 1 + 2) ) {}
?>
```

See also Operators Precedence.

9.706.1 Suggestions

- Remove useless parenthesis, unless they are important for readability.

Short name	Structures/UselessParenthesis
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Mautic*, *Woocommerce*

9.707 Useless Referenced Argument

The argument has a reference, but is only used for reading.

This is probably a development artefact that was forgotten. It is better to remove it.

This analysis also applies to foreach() loops, that declare the blind variable as reference, then use the variable as an object, accessing properties and methods. When a variable contains an object, there is no need to declare a reference : it is a reference automatically.

```
<?php

function foo($a, &$b, &$c) {
    // $c is passed by reference, but only read. The reference is useless.
    $b = $c + $a;
    // The reference is useful for $b
}

foreach ($array as &$element) {
    $element->method();
}

?>
```

See also Objects and references.

9.707.1 Suggestions

- Remove the useless & from the argument
- Make an actual use of the argument before the end of the method

Short name	Functions/UselessReferenceArgument
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*Woocommerce*, *Magento*

9.708 Useless Return

The spotted functions or methods have a return statement, but this statement is useless. This is the case for constructor and destructors, whose return value are ignored or inaccessible.

When return is void, and the last element in a function, it is also useless.

```
<?php

class foo {
    function __construct() {
        // return is not used by PHP
        return 2;
    }
}

function bar(&$a) {
    $a++;
    // The last return, when empty, is useless
    return;
}

?>
```

9.708.1 Suggestions

- Remove the return expression. Keep any other calculation.

Short name	Functions/UselessReturn
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	Very high
Examples	*ThinkPHP*, *Vanilla*

9.709 Useless Switch

This switch has only one case. It may very well be replaced by a ifthen structure.

```
<?php
switch($a) {
    case 1:
        doSomething();
```

(continues on next page)

(continued from previous page)

```
        break;
}

// Same as

if ($a == 1) {
    doSomething();
}
?>
```

9.709.1 Suggestions

- Turn the switch into a if/then for better readability
- Add other cases to the switch, making it adapted to the situation

Short name	Structures/UselessSwitch
Rulesets	*Analyze*
Severity	Major
Time To Fix	Instant (5 mins)
Examples	*Phpdocumentor*, *Dolphin*

9.710 Useless Type Check

With typehint, some checks on the arguments are now handled by the type system.

In particular, a type hinted argument can't be null, unless it is explicitly nullable, or has a `null` value as default.

```
<?php

// The test on null is useless, it will never happen
function foo(A $a) {
    if (is_null($a)) {
        // do something
    }
}

// Either nullable ? is too much, either the default null is
function barbar(?A $a = null) {
}

// The test on null is useful, the default value null allows it
function bar(A $a = null) {
    if ($a === null) {
        // do something
    }
}


?>
```

See also Type Declarations.

9.710.1 Suggestions

- Remove the nullable typehint
- Remove the null default value
- Remove tests on null

Short name	Functions/UselessTypeCheck
Rulesets	*Dead code*
Severity	Minor
Time To Fix	Quick (30 mins)

9.711 Useless Typehint

__get and __set magic methods won't use any typehint. The name of the magic property is always cast to string.

__call()

```
<?php

class x {
    // typehint is set and ignored
    function __set(float $name, string $value) {
        $this->$name = $value;
    }

    // typehint is set and ignored
    function __get(integer $name) {
        $this->$name = $value;
    }

    // typehint is checked by PHP 8.0 linting
    // typehint is enforced by PHP 7.x
    function __call(integer $name) {
        $this->$name = $value;
    }
}

$o = new x;
$b = array();
// Property will be called 'Array'
$o->{$b} = 2;

// type of $m is check at calling time. It must be string.
$o->{$m}();

?>
```

See also __set.

9.711.1 Suggestions

- Use *string* for the *$name* parameter

- Use no typehint for the *$name* parameter

Short name	Classes/UselessTypehint
Rulesets	*Suggestions*, *ClassReview*
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

9.712 Useless Unset

There are situations where trying to remove a variable is actually useless.

PHP ignores any command that tries to unset a global variable, a static variable, or a blind variable from a foreach loop.

This is different from the garbage collector, which is run on its own schedule. It is also different from an explicit unset, aimed at freeing memory early : those are useful.

```
<?php

function foo($a) {
    // unsetting arguments is useless
    unset($a);

    global $b;
    // unsetting global variable has no effect
    unset($b);

    static $c;
    // unsetting static variable has no effect
    unset($c);

    foreach($d as &$e){
        // unsetting a blind variable is useless
        (unset) $e;
    }
    // Unsetting a blind variable AFTER the loop is good.
    unset($e);
}

?>
```

See also unset.

9.712.1 Suggestions

- Remove the unset
- Set the variable to null : the effect is the same on memory, but the variable keeps its existence.
- Omit unsetting variables, and wait for the end of the scope. That way, PHP free memory en mass.

Short name	Structures/UselessUnset
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
ClearPHP	no-useless-unset
Examples	*Tine20*, *Typo3*

9.713 Uses Default Values

Default values are provided to methods so as to make it convenient to use. However, with new versions, those values may change. For example, in PHP 5.4, htmlentities() switched from `Latin1` to `UTF-8` default encoding.

```
<?php

$string = Eu não sou o pão;

echo htmlentities($string);

// PHP 5.3 : Eu n&Atilde;&pound;o sou o p&Atilde;&pound;o
// PHP 5.4 : Eu n&atilde;o sou o p&atilde;o

// Stable across versions
echo htmlentities($string, 'UTF8');

?>
```

As much as possible, it is recommended to use explicit values in those methods, so as to prevent from being surprise at a future PHP evolution.

This analyzer tend to report a lot of false positives, including usage of count(). Count() indeed has a second argument for recursive counts, and a default value. This may be ignored safely.

9.713.1 Suggestions

- Mention all arguments, as much as possible

Short name	Functions/UsesDefaultArguments
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.714 Using $this Outside A Class

`$this` is a special variable, that should only be used in a class context.

Until PHP 7.1, `$this` may be used as an argument in a function or a method, a global, a static : while this is legit, it sounds confusing enough to avoid it.

```php
<?php

function foo($this) {
    echo $this;
}

// A closure can be bound to an object at later time. It is valid usage.
$closure = function ($x) {
    echo $this->foo($x);
}

?>
```

Starting with PHP 7.1, the PHP engine check thoroughly that `$this` is used in an appropriate manner, and raise fatal errors in case it isn't.

Yet, it is possible to find `$this` outside a class : if the file is included inside a class, then `$this` will be recognized and validated. If the file is included outside a class context, it will yield a fatal error : ``Using `$this <https://www.php.net/manual/en/language.oop5.basic.php>`_ when not in object context.``

See also Closure::bind and The Basics.

Short name	Classes/UsingThisOutsideAClass
Rulesets	*Analyze*, *CompatibilityPHP71*, *LintButWontExec*
Php Version	With PHP 7.0 and older
Severity	Critical
Time To Fix	Instant (5 mins)

9.715 Using Deprecated Method

A call to a deprecated method has been spotted. A method is deprecated when it bears a `@deprecated` parameter in its typehint definition.

Deprecated methods which are not called are not reported.

```php
<?php

// not deprecated method
not_deprecated();

// deprecated method
deprecated();

/**
 * @deprecated since version 2.0.0
 */
function deprecated() {}

function not_deprecated() {}

?>
```

See also @deprecated.

9.715.1 Suggestions

- Replace the deprecated call with a stable call

Short name	Functions/UsingDeprecated
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.716 Usort Sorting In PHP 7.0

Usort(), uksort() and uasort() behavior has changed in PHP 7. Values that are equals (based on the user-provided method) may be sorted differently than in PHP 5.

If this sorting is important, it is advised to add extra comparison in the user-function and avoid returning 0 (thus, depending on default implementation).

```
<?php

$a = [ 2, 4, 3, 6];

function noSort($a) { return $a > 5; }

usort($a, 'noSort');
print_r($a);

?>
```

In PHP 5, the results is ::

```
Array
(
    [0] => 3
    [1] => 4
    [2] => 2
    [3] => 6
)
```

in PHP 7, the result is ::

```
Array
(
    [0] => 2
    [1] => 4
    [2] => 3
    [3] => 6
)
```

Short name	Php/UsortSorting
Rulesets	*CompatibilityPHP70*
Severity	Major
Time To Fix	Slow (1 hour)

9.717 Var Keyword

Var was used in PHP 4 to mark properties as public. Nowadays, new keywords are available : public, protected, private. Var is equivalent to public.

It is recommended to avoid using var, and explicitly use the new keywords.

```php
<?php

class foo {
    public $bar = 1;
    // Avoid var
    //var $bar = 1;
}

?>
```

See also Visibility.

9.717.1 Suggestions

- It is recommended to avoid using var, and explicitly use the new keywords : private, protected, public

Short name	Classes/OldStyleVar
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
ClearPHP	no-php4-class-syntax
Examples	*xataface*

9.718 Variable Global

Variable global such are valid in PHP 5.6, but no in PHP 7.0. They should be replaced with ${$foo->bar}.

```php
<?php

// Forbidden in PHP 7
global $normalGlobal;

// Forbidden in PHP 7
global $$variable->global ;

// Tolerated in PHP 7
global ${$variable->global};

?>
```

Short name	Structures/VariableGlobal
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and older
Severity	Minor
Time To Fix	Quick (30 mins)

9.719 Variable Is Not A Condition

Avoid using a lone variable as a condition. It is recommended to use a comparative value, or one of the filtering function, such as isset(), empty().

Using the raw variable as a condition blurs the difference between an undefined variable and an empty value. By using an explicit comparison or validation function, it is easier to understand what the variable stands for.

```
<?php

if (isset($error)) {
    echo 'Found one error : '.$error!;
}

//
if ($errors) {
    print count($errors).' errors found : '.join('', $errors).PHP_EOL;
    echo 'Not found';
}

?>
```

Thanks to the PMB team for the inspiration.

9.719.1 Suggestions

- Make the validation explicit, by using a comparison operator, or one of the validation function.

Short name	Structures/NoVariableIsACondition
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.720 Variables With One Letter Names

Variables with one letter name are the shortest name for variables. They also bear very little meaning : what does contain the variable `$w` ?

Some one-letter variables have meaning : `$x` and `$y` for coordinates, `$i`, `$j`, `$k` for blind variables. Others tend to be an easy way to give a name to a variable, without thinking too hard a good name.

```
<?php

// $a is reported as a one-letter variable
$a = 0;

// $i is considered a false positive.
for($i = 0; $i < 10; ++$i) {
    $a += doSomething($i);
}

?>
```

See also Using single characters for variable names in loops/exceptions and Single Letter Variable Names Still Considered Harmful.

9.720.1 Suggestions

- Make the variable more meaningful, with full words

Short name	Variables/VariableOneLetter
Rulesets	*Semantics*
Precision	Very high

9.721 Weak Typing

The test on a variable is not enough. The variable is checked for null, then used as an object or an array.

```php
<?php

if ($a !== null) {
    echo $a->b;
}

?>
```

See also From assumptions to assertions.

9.721.1 Suggestions

- Use instanceof when checking for objects
- Use is_array() when checking for arrays. Also consider is_string(), is_int(), etc.
- Use typehint when the variable is an argument

Short name	Classes/WeakType
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*TeamPass*

9.722 Weird Array Index

Array index that looks weird. Arrays index may be string or integer, but some strings looks weird.

In particular, strings that include terminal white spaces, often leads to missed values.

```php
<?php

$array = ['a ' => 1, 'b' => 2, 'c' => 3];

// Later in the code
```

(continues on next page)

(continued from previous page)

```
//Notice: Undefined index: a in /Users/famille/Desktop/analyzeG3/test.php on line 8
echo $array['a'];

//Notice: Undefined index: b  in /Users/famille/Desktop/analyzeG3/test.php on line 10
// Note that the space is visible, but easy to miss
echo $array['b '];

// all fine here
echo $array['c'];

?>
```

Although this is rare error, and often easy to spot, it is also very hard to find when it strikes.

9.722.1 Suggestions

- Remove white spaces when using strings as array index.

Short name	Arrays/WeirdIndex
Rulesets	*Semantics*
Severity	Minor
Time To Fix	Quick (30 mins)

9.723 While(List() = Each())

This code structure is quite old : it should be replace by the more modern and efficient foreach.

This structure is deprecated since PHP 7.2. It may disappear in the future.

```
<?php

    while(list($key, $value) = each($array)) {
        doSomethingWith($key) and $value();
    }

    foreach($array as $key => $value) {
        doSomethingWith($key) and $value();
    }
?>
```

See also PHP RFC: Deprecations for PHP 7.2 : ‘Each() <https://wiki.php.net/rfc/deprecations_php_7_2#each>‘_.

9.723.1 Suggestions

- Change this loop with foreach
- Change this loop with an array_* function with a callback

Short name	Structures/WhileListEach
Rulesets	*Analyze*, *Performances*, *Suggestions*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*OpenEMR*, *Dolphin*

9.724 Written Only Variables

Those variables are being written, but never read. This way, they are useless and should be removed, or read at some point.

```
<?php

// $a is used multiple times, but never read
$a = 'a';
$a .= 'b';

$b = 3;
//$b is actually read once
$a .= $b + 3;

?>
```

9.724.1 Suggestions

- Check that variables are written AND read in each context
- Remove variables that are only read
- Use the variable that are only read

Short name	Variables/WrittenOnlyVariable
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Slow (1 hour)
ClearPHP	no-unused-variable
Examples	*Dolibarr*, *SuiteCrm*

9.725 Wrong Access Style to Property

Use the right syntax when reaching for a property. Static properties use the `\:\:` operator, and non-static properties use `->`.

Mistaking one of the other raise two different reactions from PHP : `Access to undeclared `static <https://www.php.net/manual/en/language.oop5.static.php>`_ property` is a fatal error, while `PHP Notice: Accessing `static <https://www.php.net/manual/en/language.oop5.static.php>`_ property aa\:\:$a as non `static <https://www.php.net/manual/en/language.oop5.static.php>`_` is a notice.

```
<?php

class a {
    static public $a = 1;

    function foo() {
        echo self::$a; // right
        echo $this->a; // WRONG
    }
}

class b {
    public $b = 1;

    function foo() {
        echo $this->$b;  // right
        echo b::$b;      // WRONG
    }
}

?>
```

This analysis reports both static properties with a -> access, and non-static properties with a :: access.

See also Static Keyword <https://www.php.net/manual/en/language.oop5.‘static.php>‘_.

9.725.1 Suggestions

- Match the property call with the definition
- Make the property static

Short name	Classes/UndeclaredStaticProperty
Rulesets	*Analyze*, *ClassReview*, *CI-checks*
Severity	Critical
Time To Fix	Quick (30 mins)
Examples	*HuMo-Gen*

9.726 Wrong Argument Type

Checks that the type of the argument is consistent with the type of the called method.

```
<?php

function foo(int $a) { }

//valid call, with an integer
foo(1);

//invalid call, with a string
foo('asd');

?>
```

This analysis is valid with PHP 8.0.

9.726.1 Suggestions

- Always use a valid type when calling methods.

Short name	Functions/WrongArgumentType
Rulesets	*Analyze*, *Typechecks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.727 Wrong Case Namespaces

Namespaces are case-insentives.

```
<?php

// Namespaces should share the same case
namespace X {}

namespace x {}

?>
```

9.727.1 Suggestions

- Synchronize all names

Short name	Namespaces/WrongCase
Rulesets	none
Severity	Minor
Time To Fix	Quick (30 mins)

9.728 Wrong Class Name Case

The spotted classes are used with a different case than their definition. While PHP accepts this, it makes the code harder to read.

It may also be a violation of coding conventions.

```
<?php

// This use statement has wrong case for origin.
use Foo as X;

// Definition of the class
class foo {}

// Those instantiations have wrong case
```

(continues on next page)

(continued from previous page)

```
new FOO();
new X();

?>
```

See also PHP class name constant case sensitivity and PSR-11.

9.728.1 Suggestions

- Match the defined class name with the called name

Short name	Classes/WrongCase
Rulesets	*Coding Conventions*, *Coding Conventions*, *Coding Conventions*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*WordPress*

9.729 Wrong Function Name Case

The spotted functions are used with a different case than their definition. While PHP accepts this, it makes the code harder to read.

It may also be a violation of coding conventions.

```
<?php

// Definition of the class
function foo () {}

// Those calls have wrong case
FOO();
\Foo();

// This is valid
foo();

?>
```

See also PHP class name constant case sensitivity and PSR-11.

9.729.1 Suggestions

- Match the defined functioncall with the called name

Short name	Functions/WrongCase
Rulesets	none
Severity	Minor
Time To Fix	Instant (5 mins)

9.730 Wrong Number Of Arguments

Those functioncalls are made with too many or too few arguments.

When the number arguments is wrong for native functions, PHP emits a warning. When the number arguments is too small for custom functions, PHP raises an exception. When the number arguments is too high for custom functions, PHP ignores the arguments. Such arguments should be handled with the variadic operator, or with func_get_args() family of functions.

```
<?php

echo strtoupper('This function is', 'ignoring arguments');
//Warning: strtoupper() expects exactly 1 parameter, 2 given in Command line code on␣
↪line 1

echo strtoupper();
//Warning: strtoupper() expects exactly 1 parameter, 0 given in Command line code on␣
↪line 1

function foo($argument) {}
echo foo();
//Fatal error: Uncaught ArgumentCountError: Too few arguments to function foo(), 0␣
↪passed in /Users/famille/Desktop/analyzeG3/test.php on line 10 and exactly 1␣
↪expected in /Users/famille/Desktop/analyzeG3/test.php:3

echo foo('This function is', 'ignoring arguments');

?>
```

It is recommended to check the signature of the methods, and fix the arguments.

9.730.1 Suggestions

- Add more arguments to fill the list of compulsory arguments
- Remove arguments to fit the list of compulsory arguments
- Use another method or class

Short name	Functions/WrongNumberOfArguments
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-missing-argument.md
Examples	*xataface*

9.731 Wrong Optional Parameter

Wrong placement of optional parameters.

PHP parameters are optional when they defined with a default value, like this :

```
<?php
    function x($arg = 1) {
        // PHP code here
    }
?>
```

When a function have both compulsory and optional parameters, the compulsory ones should appear first, and the optional should appear last :

```
<?php
    function x($arg, $arg2 = 2) {
        // PHP code here
    }
?>
```

PHP solves this problem at runtime, assign values in the same other, but will miss some of the default values and emits warnings.

It is better to put all the optional parameters at the end of the method's signature.

Optional parameter wrongly placed are now a Notice in PHP 8.0. The only previous case that is allowed in PHP 8.0 and also in this analysis, is when the `null` value is used as default for typed arguments.

See also Function arguments.

9.731.1 Suggestions

- Give default values to all but first parameters. Null is a good default value, as PHP will use it if not told otherwise.
- Remove default values to all but last parameters. That is probably a weak solution.
- Change the order of the values, so default-valued parameters are at the end. This will probably have impact on the rest of the code, as the API is changing.

Short name	Functions/WrongOptionalParameter
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*FuelCMS*, *Vanilla*

9.732 Wrong Parameter Type

The expected parameter is not of the correct type. Check PHP documentation to know which is the right format to be used.

```
<?php

// substr() shouldn't work on integers.
// the first argument is first converted to string, and it is 123456.
echo substr(123456, 0, 4); // display 1234

// substr() shouldn't work on boolean
// the first argument is first converted to string, and it is 1, and not t
```

(continues on next page)

(continued from previous page)

```
echo substr(true, 0, 1); // displays 1

// substr() works correctly on strings.
echo substr(123456, 0, 4);

?>
```

Short name	Php/InternalParameterType
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Zencart*

9.733 Wrong Range Check

The interval check should use && and not ||.

```
<?php

//interval correctly checked a is between 2 and 999
if ($a > 1 && $a < 1000) {}

//interval incorrectly checked : a is 2 or more ($a < 1000 is never checked)
if ($a > 1 || $a < 1000) {}

?>
```

9.733.1 Suggestions

- Make the interval easy to read and understand
- Check the truth table for the logical operation

Short name	Structures/WrongRange
Rulesets	*Analyze*
Severity	Major
Time To Fix	Instant (5 mins)
Examples	*Dolibarr*, *WordPress*

9.734 Wrong Returned Type

The returned value is not compatible with the specified return type.

```
<?php

// classic error
function bar() : int {
    return 'A';
```

(continues on next page)

(continued from previous page)

```
}

// classic static error
const B = 2;
function bar() : string {
    return B;
}

// undecideable error
function bar($c) : string {
    return $c;
}

// PHP lint this, but won't execute it
function foo() : void {
    // No return at all
}

?>
```

See also Returning values and Void Return Type.

9.734.1 Suggestions

- Match the return type with the return value
- Remove the return expression altogether
- Add a typecast to the returning expression

Short name	Functions/WrongReturnedType
Rulesets	*Analyze*, *ClassReview*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.735 Wrong Type For Native PHP Function

This analysis reports calls to a PHP native function with a wrongly typed value.

```
<?php

// valid calls
echo exp(1);
echo exp(2.5);

// invalid calls
echo exp(1);
echo exp(array(2.5));

// valid call, but invalid math
// -1 is not a valid value for log(), but -1 is a valid type (int) : it is not␣
↪reported by this analysis.
```

(continues on next page)

(continued from previous page)

```
echo log(-1);
?>
```

9.735.1 Suggestions

- Set the code to the valid type, when calling a PHP native function

Short name	Php/WrongTypeForNativeFunction
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.736 Wrong Type With Call

This analysis checks that a call to a method uses the right literal values' types.

Currently, this analysis doesn't take into account `strict_types = 1`.

```
<?php

function foo(string $a) {

}

// wrong type used
foo(1);

// wrong type used
foo("1");

?>
```

9.736.1 Suggestions

- Use the right type with all literals

Short name	Functions/WrongTypeWithCall
Rulesets	*Analyze*, *Typechecks*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

9.737 Wrong Typed Property Default

Property is typed with an incompatible default value type.

Init type might be a new instance, the return of a method call or an interface compatible object.

```
<?php

class x {
    private A $property;
    private B $incompatible;

    function __construct() {
        // This is compatible
        $this->property = new A();

        // This is incompatible : new B() expected
        $this->incompatible = new C();

    }
}

?>
```

PHP compiles such code, but won't execute it, as it detects the incompatibility.

9.737.1 Suggestions

- Remove the type hint of the property
- Fix the initialization call
- Use an interface for typehint

Short name	Classes/WrongTypedPropertyInit
Rulesets	*Analyze*, *LintButWontExec*, *ClassReview*, *CI-checks*
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)

9.738 Wrong Typehinted Name

The parameter name doesn't reflect the typehint used.

There are no restriction on parameter names, except its uniqueness in the signature. Yet, using a scalar typehint as the name for another typehinted value is just misleading.

```
<?php

function foo(string $array,
             int $int) {
    // doSomething()
}

function bar(array $strings) {
    // doSomething()
}

?>
```

The comparison relies on exact names : calling an array a list of `strings` is OK with this analysis.

9.738.1 Suggestions

- Rename

Short name	Functions/WrongTypehintedName
Rulesets	*Coding Conventions*, *Semantics*
Severity	Minor
Time To Fix	Quick (30 mins)

9.739 Wrong fopen() Mode

Wrong file opening for fopen().

fopen() has a few modes, as described in the documentation : 'r', 'r+', for reading; 'w', 'w+' for writing; 'a', 'a+' for appending; 'x', 'x+' for modifying; 'c', 'c+' for writing and locking, 't' for text files and windows only. An optional 'b' may be used to make the fopen() call more portable and binary safe. Another optional 't' may be used to make the fopen() call process automatically text input : this one should be avoided.

```
<?php

// open the file for reading, in binary mode
$fp = fopen('/tmp/php.txt', 'rb');

// New option e in PHP 7.0.16 and 7.1.2 (beware of compatibility)
$fp = fopen('/tmp/php.txt', 'rbe');

// Unknown option x
$fp = fopen('/tmp/php.txt', 'rbx');

?>
```

Any other values are not understood by PHP.

9.739.1 Suggestions

- Check the docs, choose the right opening mode.

Short name	Php/FopenMode
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Tikiwiki*, *HuMo-Gen*

9.740 Yoda Comparison

Yoda comparison is a way to write conditions which places literal values on the left side.

```
<?php
  if (1 == $a) {
    // Then condition
  }
?>
```

The objective is to avoid mistaking a comparison to an assignation. If the comparison operator is mistaken, but the literal is on the left, then an error will be triggered, instead of a silent bug.

```
<?php
    // error in comparison!
    if ($a = 1) {
        // Then condition
    }
?>
```

See also Yoda Conditions, Yoda Conditions: To Yoda or Not to Yoda.

Short name	Structures/YodaComparison
Rulesets	*Coding Conventions*
Severity	Minor
Time To Fix	Slow (1 hour)

9.741 __DIR__ Then Slash

__DIR__ must be concatenated with a string starting with /.

The magic constant __DIR__ holds the name of the current directory, without final /. When it is used to build path, then the following path fragment must start with /. Otherwise, two directories names will be merged together.

```
<?php

// __DIR__ = /a/b/c
// $filePath = /a/b/c/g.php

// /a/b/c/d/e/f.txt : correct path
echo __DIR__.'/d/e/f.txt';
echo dirname($filePath).'/d/e/f.txt';

// /a/b/cd/e/f.txt : most probably incorrect path
echo __DIR__.'d/e/f.txt';
echo dirname($filePath).'d/e/f.txt';

?>
```

9.741.1 Suggestions

- Add a check on __DIR__, as it may be '/' when run at the root of the server
- Add a '/' at the beginning of the path after __DIR__.
- Add a call to realpath() or file_exists(), before accessing the file.

Short name	Structures/DirThenSlash
Rulesets	*Analyze*, *CI-checks*
Severity	Major
Time To Fix	Instant (5 mins)
Examples	*Traq*

9.742 __debugInfo() Usage

The magic method __debugInfo() provides a custom way to dump an object.

It has been introduced in PHP 5.6. In the previous versions of PHP, this method is ignored and won't be called when debugging.

```
<?php

// PHP 5.6 or later
class foo {
    private $bar = 1;
    private $reallyHidden = 2;

    function __debugInfo() {
        return ['bar' => $this->bar,
                'reallyHidden' => 'Secret'];
    }
}

$f = new Foo();
var_dump($f);

/* Displays :
object(foo)#1 (2) {
  [bar]=>
  int(1)
  [reallyHidden]=>
  string(6) Secret
}
*/

?>
```

See also Magic methods.

Short name	Php/debugInfoUsage
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Php Version	With PHP 5.6 and more recent
Severity	Minor
Time To Fix	Slow (1 hour)
Examples	*Dolibarr*

9.743 __toString() Throws Exception

Magical method __toString() can't throw exceptions.

In fact, __toString() may not let an exception pass. If it throw an exception, but must catch it. If an underlying method throws an exception, it must be caught.

```
<?php

class myString {
    private $string = null;

    public function __construct($string) {
        $this->string = $string;
    }

    public function __toString() {
        // Do not throw exceptions in __toString
        if (!is_string($this->string)) {
            throw new Exception("$this->string is not a string!!");
        }

        return $this->string;
    }
}

?>
```

A fatal error is displayed, when an exception is not intercepted in the __toString() function.

::

> PHP Fatal error: Method myString::__toString() must not throw an exception, caught Exception: 'Exception message' in `file.php`

See also __toString().

9.743.1 Suggestions

- Remove any usage of exception from __toString() magic method

Short name	Structures/toStringThrowsException
Rulesets	*Analyze*
Php Version	7.4-
Severity	Major
Time To Fix	Quick (30 mins)

9.744 array_key_exists() Speedup

isset() used to be the fastest, but array_key_exists() is. Since PHP 7.4, array_key_exists() has its own opcode, leading to better features and speed.

isset() is faster for all non-empty values, but is limited when the value is NULL or empty : then, array_key_exists() has the good features.

```
This change makes `array_key_exists() <https://www.php.net/array_key_exists>`_
actually faster than `isset() <https://www.www.php.net/isset>`_ by ~25%
(tested with GCC 8, -O3, march=native, mtune=native)..
```

```
<?php

$foo = [123 => 456];

// This is sufficient and efficient since PHP 7.4
if (array_search_key($foo[123])) {
    // do something
}

// taking advantages of performances for PHP 7.4 and older
if (isset($foo[123]) || array_search_key($foo[123])) {
    // do something
}

?>
```

See also Implement ZEND_ARRAY_KEY_EXISTS opcode to speed up ‘array_key_exists() <https://github.com/php/php-src/pull/3360>‘_.

9.744.1 Suggestions

- Remove the logical test and the isset() call

Short name	Performances/ArrayKeyExistsSpeedup
Rulesets	*Suggestions*, *Performances*
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)

9.745 array_key_exists() Works On Arrays

array_key_exists() requires arrays as second argument. Until PHP 7.4, objects were also allowed, yet it is now deprecated.

```
<?php

// Valid way to check for key
$array = ['a' => 1];
var_dump(array_key_exists('a', $array))


// Deprecated since PHP 7.4
$object = new Stdclass();
$object->a = 1;
var_dump(array_key_exists('a', $object))

?>
```

See also array_key_exists() with objects, and array_key_exists, and.

9.745.1 Suggestions

- Use the (array) cast to turn the object into an array
- Use the native PHP function proprety_exists() or isset() on the property to check them.

Short name	Php/ArrayKeyExistsWithObjects
Rulesets	*CompatibilityPHP74*, *Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)

9.746 array_merge() And Variadic

Always check value in variadic before using it with array_merge() and array_merge_recursive().

Before PHP 7.4, array_merge() and array_merge_recursive() would complain when no argument was provided. As such, using the spread operator ... on an empty array() would yield no argument, and an error.

```
<?php

//
$b = array_merge(...$x);



?>
```

9.746.1 Suggestions

- Add a check to the spread variable to ensure it is not empty
- Append an empty array to to the spread variable to ensure it is not empty

Short name	Structures/ArrayMergeAndVariadic
Rulesets	*Analyze*
Php Version	7.4-
Severity	Minor
Time To Fix	Quick (30 mins)

9.747 crypt() Without Salt

PHP requires a salt when calling crypt(). 5.5 and previous versions didn't require it. Salt is a simple string, that is usually only known by the application.

According to the manual : The salt parameter is optional. However, crypt() creates a weak hash without the salt. PHP 5.6 or later raise an E_NOTICE error without it. Make sure to specify a strong enough salt for better security.

```
<?php
// Set the password
$password = 'mypassword';
```

(continues on next page)

(continued from previous page)

```

// salted crypt usage (always valid)
$hash = crypt($password, '123salt');

// Get the hash, letting the salt be automatically generated
// This generates a notice after PHP 5.6
$hash = crypt($password);

?>
```

See also crypt.

9.747.1 Suggestions

- Always provide the second argument

Short name	Structures/CryptWithoutSalt
Rulesets	*CompatibilityPHP54*
Php Version	With PHP 5.6 and older
Severity	Minor
Time To Fix	Instant (5 mins)

9.748 curl_version() Has No Argument

curl_version() used to accept `CURLVERSION_NOW` as argument. Since PHP 7.4, it is a function without arguments.

```
<?php

// Compatible syntax
$details = curl_version(CURLVERSION_NOW);

// New PHP 7.4 syntax
$details = curl_version();

?>
```

See also curl_version.

9.748.1 Suggestions

- Drop all arguments from curl_version() calls.

Short name	Structures/CurlVersionNow
Rulesets	*CompatibilityPHP74*
Severity	Minor
Time To Fix	Quick (30 mins)

9.749 error_reporting() With Integers

Using named constants with error_reporting is strongly encouraged to ensure compatibility for future versions. As error levels are added, the range of integers increases, so older integer-based error levels will not always behave as expected. (Adapted from the documentation).

```
<?php

// This is ready for PHP next version
error_reporting(E_ALL & ~E_DEPRECATED & ~E_STRICT & ~E_NOTICE & ~E_WARNING);

// This is not ready for PHP next version
error_reporting(2047);

// -1 and 0 are omitted, as they will be valid even is constants changes.
error_reporting(-1);
error_reporting(0);

?>
```

See also directive error_reporting and error_reporting.

9.749.1 Suggestions

- Always use the constant combination when configuring error_reporting or any PHP native function

Short name	Structures/ErrorReportingWithInteger
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*SugarCrm*

9.750 eval() Without Try

`eval()` emits a `ParseError` exception with PHP 7 and later. Catching this exception is the recommended way to handle errors when using the `eval()` function.

```
<?php

$code = 'This is no PHP code.';

//PHP 5 style
eval($code);
// Ends up with a Fatal error, at execution time

//PHP 7 style
try {
    eval($code);
} catch (ParseError $e) {
    cleanUpAfterEval();
}

?>
```

Note that it will catch situations where `eval()` is provided with code that can't be used, but it will not catch security problems. Avoid using `eval()` with incoming data.

9.750.1 Suggestions

- Always add a try/catch block around eval() call

Short name	Structures/EvalWithoutTry
Rulesets	*Analyze*, *Security*, *CI-checks*
Php Version	With PHP 7.0 and more recent
Severity	Critical
Time To Fix	Quick (30 mins)
Examples	*FuelCMS*, *ExpressionEngine*

9.751 ext/apc

Extension Alternative PHP Cache.

The Alternative PHP Cache (APC) is a free and open opcode cache for PHP. Its goal is to provide a free, open, and robust framework for caching and optimizing PHP intermediate code.

This extension is considered unmaintained and dead.

```php
<?php
   $bar = 'BAR';
   apc_add('foo', $bar);
   var_dump(apc_fetch('foo'));
   echo PHP_EOL;

   $bar = 'NEVER GETS SET';
   apc_add('foo', $bar);
   var_dump(apc_fetch('foo'));
   echo PHP_EOL;
?>
```

See also Alternative PHP Cache.

Short name	Extensions/Extapc
Rulesets	*CompatibilityPHP55*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.752 ext/dba

Extension ext/dba.

These functions build the foundation for accessing Berkeley DB style databases.

```
<?php

$id = dba_open('/tmp/test.db', 'n', 'db2');

if (!$id) {
    echo 'dba_open failed'.PHP_EOL;
    exit;
}

dba_replace('key', 'This is an example!', $id);

if (dba_exists('key', $id)) {
    echo dba_fetch('key', $id);
    dba_delete('key', $id);
}

dba_close($id);
?>
```

See also Database (dbm-style) Abstraction Layer.

Short name	Extensions/Extdba
Rulesets	*CompatibilityPHP53*
Severity	Major
Time To Fix	Slow (1 hour)

9.753 ext/ereg

Extension ext/ereg.

```
<?php
if (ereg ('([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})', $date, $regs)) {
    echo $regs[3].'.'.$regs[2].'.'.$regs[1];
} else {
    echo 'Invalid date format: '.$date;
}
?>
```

See also Ereg.

Short name	Extensions/Extereg
Rulesets	*CompatibilityPHP70*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Quick (30 mins)

9.754 ext/fdf

Extension ext/fdf.

Forms Data Format (FDF) is a format for handling forms within PDF documents.

```
<?php
$outfdf = fdf_create();
fdf_set_value($outfdf, 'volume', $volume, 0);

fdf_set_file($outfdf, 'http:/testfdf/resultlabel.pdf');
fdf_save($outfdf, 'outtest.fdf');
fdf_close($outfdf);
Header('Content-type: application/vnd.fdf');
$fp = fopen('outtest.fdf', 'r');
fpassthru($fp);
unlink('outtest.fdf');
?>
```

See also Form Data Format.

Short name	Extensions/Extfdf
Rulesets	*CompatibilityPHP53*
Severity	Major
Time To Fix	Slow (1 hour)

9.755 ext/mcrypt

Extension for mcrypt.

This extension has been deprecated as of PHP 7.1.0 and moved to PECL as of PHP 7.2.0.

This is an interface to the mcrypt library, which supports a wide variety of block algorithms such as DES, TripleDES, Blowfish (default), 3-WAY, SAFER-SK64, SAFER-SK128, TWOFISH, TEA, RC2 and GOST in CBC, OFB, CFB and ECB cipher modes. Additionally, it supports RC6 and IDEA which are considered 'non-free'. CFB/OFB are 8bit by default.

```
<?php
    # --- ENCRYPTION ---

    # the key should be random binary, use scrypt, bcrypt or PBKDF2 to
    # convert a string into a key
    # key is specified using hexadecimal
    $key = pack('H*',
↪'bcb04b7e103a0cd8b54763051cef08bc55abe029fdebae5e1d417e2ffb2a00a3');

    # show key size use either 16, 24 or 32 byte keys for AES-128, 192
    # and 256 respectively
    $key_size =  strlen($key);
    echo 'Key size: ' . $key_size . PHP_EOL;

    $plaintext = 'This string was AES-256 / CBC / ZeroBytePadding encrypted.';

    # create a random IV to use with CBC encoding
    $iv_size = mcrypt_get_iv_size(MCRYPT_RIJNDAEL_128, MCRYPT_MODE_CBC);
    $iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);

    # creates a cipher text compatible with AES (Rijndael block size = 128)
    # to keep the text confidential
    # only suitable for encoded input that never ends with value 00h
```

(continues on next page)

(continued from previous page)

```
    # (because of default zero padding)
    $ciphertext = mcrypt_encrypt(MCRYPT_RIJNDAEL_128, $key,
                                 $plaintext, MCRYPT_MODE_CBC, $iv);

    # prepend the IV for it to be available for decryption
    $ciphertext = $iv . $ciphertext;

    # encode the resulting cipher text so it can be represented by a string
    $ciphertext_base64 = base64_encode($ciphertext);

    echo  $ciphertext_base64 . PHP_EOL;

    # === WARNING ===

    # Resulting cipher text has no integrity or authenticity added
    # and is not protected against padding oracle attacks.

    # --- DECRYPTION ---

    $ciphertext_dec = base64_decode($ciphertext_base64);

    # retrieves the IV, iv_size should be created using mcrypt_get_iv_size()
    $iv_dec = substr($ciphertext_dec, 0, $iv_size);

    # retrieves the cipher text (everything except the $iv_size in the front)
    $ciphertext_dec = substr($ciphertext_dec, $iv_size);

    # may remove 00h valued characters from end of plain text
    $plaintext_dec = mcrypt_decrypt(MCRYPT_RIJNDAEL_128, $key,
                                    $ciphertext_dec, MCRYPT_MODE_CBC, $iv_dec);

    echo  $plaintext_dec . PHP_EOL;
?>
```

See also extension mcrypt and mcrypt.

Short name	Extensions/Extmcrypt
Rulesets	*CompatibilityPHP71*
Severity	Major
Time To Fix	Slow (1 hour)

9.756 ext/mhash

Extension mhash (obsolete since PHP 5.3.0).

This extension provides functions, intended to work with mhash.

```
<?php
$input = 'what do ya want for nothing?';
$hash = mhash(MHASH_MD5, $input);
echo 'The hash is ' . bin2hex($hash) . '<br />'.PHP_EOL;
$hash = mhash(MHASH_MD5, $input, 'Jefe');
echo 'The hmac is ' . bin2hex($hash) . '<br />'.PHP_EOL;
?>
```

See also Extension mhash.

Short name	Extensions/Extmhash
Rulesets	*CompatibilityPHP54*
Severity	Major
Time To Fix	Slow (1 hour)

9.757 ext/ming

Extension ext/ming, to create swf files with PHP.

Ming is an open-source (LGPL) library which allows you to create SWF (‘Flash’) format movies.

```
<?php
  $s = new SWFShape();
  $f = $s->addFill(0xff, 0, 0);
  $s->setRightFill($f);

  $s->movePenTo(-500, -500);
  $s->drawLineTo(500, -500);
  $s->drawLineTo(500, 500);
  $s->drawLineTo(-500, 500);
  $s->drawLineTo(-500, -500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $i->setDepth(1);
  $p->nextFrame();

  for ($n=0; $n<5; ++$n) {
    $i->rotate(-15);
    $p->nextFrame();
  }

  $m = new SWFMovie();
  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(6000, 4000);

  $i = $m->add($p);
  $i->setDepth(1);
  $i->moveTo(-500,2000);
  $i->setName('box');

  $m->add(new SWFAction('/box.x += 3;'));
  $m->nextFrame();
  $m->add(new SWFAction('gotoFrame(0); play();'));
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

See also Ming (flash) and Ming.

Short name	Extensions/Extming
Rulesets	*CompatibilityPHP53*
Severity	Major
Time To Fix	Slow (1 hour)

9.758 ext/mysql

Extension for MySQL (Original MySQL API).

This extension is deprecated as of PHP 5.5.0, and has been removed as of PHP 7.0.0. Instead, either the mysqli or PDO_MySQL extension should be used. See also the MySQL API Overview for further help while choosing a MySQL API. .. code-block:: php <?php $result = mysql_query('SELECT * WHERE 1=1'); if (!$result) { die('Invalid query: ' . mysql_error()); } ?> See also Original MySQL API and MySQL.

Short name	Extensions/Extmysql
Rulesets	*CompatibilityPHP55*
Php Version	With PHP 7.0 and older
Severity	Major
Time To Fix	Slow (1 hour)

9.759 filter_input() As A Source

The filter_input() and filter_input_array() functions access directly to `$_GET`. They represent a source for external data just like `$_GET`, `$_POST`, etc.

The main feature of filter_input() is that it is already filtered. The main drawback is that `FILTER_FLAG_NONE` is the `none` filter, and that default configuration is *FILTER_UNSAFE_RAW*.

The filter extension keeps access to the incoming data, even after the super globals, such as `$_GET`, are unset.

```php
<?php

// Removing $_GET
$_GET = [];

// with the default : FILTER_UNSAFE_RAW, this means XSS
echo filter_input(INPUT_GET, 'i');

// Same as above :
echo filter_var(_GET, 'i');

?>
```

Thanks to Frederic Bouchery for reporting this special case.

See also Data filtering.

9.759.1 Suggestions

- Use the classic $_GET, $_POST super globals, which are easier to audit.
- Use your framework's parameter access.

Short name	Security/FilterInputSource
Rulesets	*Security*
Severity	Minor
Time To Fix	Slow (1 hour)

9.760 fputcsv() In Loops

fputcsv() is slow when called on each row. It actually flushes the data to the disk each time, and that results in a inefficient dump to the disk, each call.

To speed up this process, it is recommended to dump the csv to memory first, then dump the memory to the disk, in larger chunks. Since fputcsv() works only on stream, it is necessary to use a memory stream.

```
<?php

// Speedy yet memory intensive version
$f = fopen('php://memory', 'w+');
foreach($data_source as $row) {
    // You may configure fputcsv as usual
    fputcsv($f, $row);
}
rewind($f); // Important
$fp = fopen('final.csv', 'w+');
fputs($fp, stream_get_contents($f));
fclose($fp);
fclose($f);

// Slower version
$fp = fopen('final.csv', 'w+');
foreach($data_source as $row) {
    // You may configure fputcsv as usual
    fputcsv($fp, $row);
}
fclose($fp);
?>
```

The speed improvement is significant on small rows, while it may be less significant on larger rows : with more data in the rows, the file buffer may fill up more efficiently. On small rows, the speed gain is up to 7 times.

9.760.1 Suggestions

- Use fputcsv() on a memory stream, and flush it on the disk once

Short name	Performances/CsvInLoops
Rulesets	*Performances*, *Top10*
Severity	Minor
Time To Fix	Quick (30 mins)

9.761 func_get_arg() Modified

func_get_arg() and func_get_args() used to report the calling value of the argument until PHP 7. Since PHP 7, it is reporting the value of the argument at calling time, which may have been modified by a previous instruction.

```
<?php

function x($a) {
    $a++;
    print func_get_arg(0);
}

x(0);
?>
```

This code will display 1 in PHP 7, and 0 in PHP 5.

Short name	Functions/funcGetArgModified
Rulesets	*Analyze*, *CompatibilityPHP70*
Severity	Major
Time To Fix	Quick (30 mins)

9.762 idn_to_ascii() New Default

The default parameter value of idn_to_ascii() and idn_to_utf8() is now INTL_IDNA_VARIANT_UTS46 instead of the deprecated INTL_IDNA_VARIANT_2003.

```
<?php

echo idn_to_ascii('täst.de');

?>
```

See also idn_to_ascii, idn_to_utf8 and Unicode IDNA Compatibility Processing.

9.762.1 Suggestions

- Explicitely add the second parameter to the idn_to_ascii() and idn_to_utf8() functions.

Short name	Php/IdnUts46
Rulesets	*CompatibilityPHP74*

9.763 include_once() Usage

include_once() and require_once() functions should be avoided for performances reasons.

```
<?php

// Including a library.
include 'lib/helpers.inc';

// Including a library, and avoiding double inclusion
include_once 'lib/helpers.inc';
```

(continues on next page)

(continued from previous page)

```
?>
```

Try using autoload for loading classes, or use include() or require() and make it possible to include several times the same file without errors.

9.763.1 Suggestions

- Avoid using include_once() whenever possible
- Use autoload() to load classes, and avoid loading them with include

Short name	Structures/OnceUsage
Rulesets	*Analyze*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*XOOPS*, *Tikiwiki*

9.764 isset() With Constant

Until PHP 7, it was possible to use arrays as constants, but it was not possible to test them with isset.

```
<?php
const X = [1,2,3];

if (isset(X[4])) {}
?>
```

This would yield an error : `Cannot use `isset() <https://www.www.php.net/isset>`_ on the result of an expression (you can use "null !== expression" instead)`. This is a backward incompatibility.

Short name	Structures/IssetWithConstant
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Php Version	With PHP 7.0 and more recent
Severity	Major
Time To Fix	Instant (5 mins)

9.765 list() May Omit Variables

Simply omit any unused variable in a list() call.

list() is the only PHP function that accepts to have omitted arguments. If the following code makes no usage of a listed variable, just omit it.

```
<?php
    // No need for '2', so no assignation
    list ($a, , $b) = array(1, 2, 3);
        // works with PHP 7.1 short syntax
```

(continues on next page)

(continued from previous page)

```
         [$a, , $b] = array(1, 2, 3);

    // No need for '2', so no assignation
    list ($a, $c, $b) = array(1, 2, 3);
?>
```

See also list.

9.765.1 Suggestions

- Remove the unused variables from the list call
- When the ignored values are at the beginning or the end of the array, array_slice() may be used to shorten the array.

Short name	Structures/ListOmissions
Rulesets	*Analyze*, *Suggestions*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*OpenConf*, *FuelCMS*

9.766 mb_strrpos() Third Argument

Passing the encoding as 3rd parameter to mb_strrpos() is deprecated. Instead pass a 0 offset, and encoding as 4th parameter.

```
<?php

// Finds the position of the last occurrence of of a string in a string, starting at␣
↪position 10
$extract = mb_strrpos($haystack, $needle, 10, 'utf8');

// This is the old behavior. Here, the offset will be 0, by default
$extract = mb_strrpos($haystack, $needle, 'utf8');
?>
```

See also mb_strrpos().

9.766.1 Suggestions

-

Short name	Php/Php74mbstrrpos3rdArg
Rulesets	*CompatibilityPHP74*
Severity	Minor
Time To Fix	Quick (30 mins)

9.767 mcrypt_create_iv() With Default Values

Avoid using mcrypt_create_iv() default values.

mcrypt_create_iv() used to have MCRYPT_DEV_RANDOM as default values, and in PHP 5.6, it now uses MCRYPT_DEV_URANDOM.

```
<?php
    $size = mcrypt_get_iv_size(MCRYPT_CAST_256, MCRYPT_MODE_CFB);
    // mcrypt_create_iv is missing the second argument
    $iv = mcrypt_create_iv($size);

// Identical to the line below
//    $iv = mcrypt_create_iv($size, MCRYPT_DEV_RANDOM);

?>
```

If the code doesn't have a second argument, it relies on the default value. It is recommended to set explicitly the value, so has to avoid problems while migrating.

See also mcrypt_create_iv().

Short name	Structures/McryptcreateivWithoutOption
Rulesets	*CompatibilityPHP70*
Php Version	With PHP 5.6 and older
Severity	Minor
Time To Fix	Instant (5 mins)

9.768 move_uploaded_file Instead Of copy

Always use move_uploaded_file() with uploaded files. Avoid using copy or rename with uploaded file.

move_uploaded_file() checks to ensure that the file designated by filename is a valid upload file (meaning that it was uploaded via PHP's HTTP POST upload mechanism).

```
<?php

    // $a->file was filled with $_FILES at some point
    move_uploaded_file($a->file['tmp_name'], $target);

    // $a->file was filled with $_FILES at some point
    rename($a->file['tmp_name'], $target);

?>
```

See also move_uploaded_file and Uploading Files with PHP.

9.768.1 Suggestions

- Always use move_uploaded_file()
- Extract the needed information from the file, and leave it for PHP to remove without storage

Short name	Security/MoveUploadedFile
Rulesets	*Security*
Severity	Major
Time To Fix	Quick (30 mins)

9.769 openssl_random_pseudo_byte() Second Argument

openssl_random_pseudo_byte() uses exceptions to signal an error. Since PHP 7.4, there is no need to use the second argument.

On the other hand, it is important to catch the exception that openssl_random_pseudo_byte() may emit.

```
<?php
    // PHP 7.4 way to check on random number generation
    try {
        $bytes = openssl_random_pseudo_bytes($i);
    } catch(\Exception $e) {
        die(Error while loading random number);
    }

    // Old way to check on random number generation
    $bytes = openssl_random_pseudo_bytes($i, $cstrong);
    if ($cstrong === false) {
        die(Error while loading random number);
    }
?>
```

See also openssl_random_pseudo_byte and PHP RFC: Improve 'openssl_random_pseudo_bytes() <https://wiki.php.net/rfc/improve-openssl-random-pseudo-bytes>'_.

9.769.1 Suggestions

- Skip the second argument, add a try/catch around the call to openssl_random_pseudo_bytes()

Short name	Structures/OpensslRandomPseudoByteSecondArg
Rulesets	*CompatibilityPHP74*
Severity	Minor
Time To Fix	Quick (30 mins)

9.770 parse_str() Warning

The parse_str() function parses a query string and assigns the resulting variables to the local scope. This may create a unexpected number of variables, and even overwrite the existing one.

```
<?php
  function foo( ) {
    global $a;

    echo $a;
  }
```

(continues on next page)

(continued from previous page)

```
  parse_str('a=1'); // No second parameter
  foo( );
  // displays 1
?>
```

Always use an empty variable a second parameter to parse_str(), so as to collect the incoming values, and then, filter them in that array.

9.770.1 Suggestions

- Use the second parameter when calling parse_url();
- Change to PHP 8.0 version, which made the second argument compulsory

Short name	Security/parseUrlWithoutParameters
Rulesets	*Security*
Severity	Major
Time To Fix	Slow (1 hour)
ClearPHP	know-your-variables

9.771 preg_match_all() Flag

preg_match_all() has an option to configure the structure of the results : it is either by capturing parenthesis (by default), or by result sets.

The second option is the most interesting when the following foreach() loop has to manipulate several captured strings at the same time. No need to use an index in the first array and use it in the other arrays.

```
<?php
$string = 'ababab';

// default behavior
preg_match_all('/(a)(b)/', $string, $r);
$found = '';
foreach($r[1] as $id => $s) {
    $found .= $s.$r[2][$id];
}

// better behavior
preg_match_all('/(a)(b)/', $string, $r, PREG_SET_ORDER);
$found = '';
foreach($r as $s) {
    $found .= $s[1].$s[2];
}

?>
```

The second syntax is easier to read and may be marginally faster to execute (preg_match_all() and foreach()).

9.771.1 Suggestions

- Use flags to adapt the results of preg_match_all() to your code, not the contrary.

Short name	Php/PregMatchAllFlag
Rulesets	*Suggestions*
Severity	Minor
Time To Fix	Quick (30 mins)
Examples	*FuelCMS*

9.772 preg_replace With Option e

preg_replace() supported the /e option until PHP 7.0. It allowed the use of eval()'ed expression as replacement. This has been dropped in PHP 7.0, for security reasons.

preg_replace() with /e option may be replaced with preg_replace_callback() and a closure, or preg_replace_callback_array() and an array of closures.

```
<?php

// preg_replace with /e
$string = 'abcde';

// PHP 5.6 and older usage of /e
$replaced = preg_replace('/c/e', 'strtoupper($0)', $string);

// PHP 7.0 and more recent
// With one replacement
$replaced = preg_replace_callback('/c/', function ($x) { return strtoupper($x[0]); },
↪$string);

// With several replacements, preventing multiple calls to preg_replace_callback
$replaced = preg_replace_callback_array(array('/c/' => function ($x) { return␣
↪strtoupper($x[0]); },
                                              '/[a-b]/' => function ($x) { return␣
↪strtolower($x[0]); }), $string);
?>
```

9.772.1 Suggestions

- Replace call to preg_replace() and /e with preg_replace_callback() or preg_replace_callback_array()

Short name	Structures/pregOptionE
Rulesets	*Analyze*, *CompatibilityPHP70*, *Security*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CI-checks*
Severity	Major
Time To Fix	Quick (30 mins)
Examples	*Edusoho*

9.773 self, parent, static Outside Class

self, parent and static should be called inside a class or trait. PHP lint won't report those situations.

self, parent and static may be used in a trait : their actual value will be only known at execution time, when the trait is used.

```
<?php
// In the examples, self, parent and static may be used interchangeably

// This raises a Fatal error
//Fatal error: Uncaught Error: Cannot access static:: when no class scope is active
new static();

// static calls
echo self::CONSTANTE;
echo self::$property;
echo self::method();

// as a type hint
function foo(static $x) {
    doSomething();
}

// as a instanceof
if ($x instanceof static) {
    doSomething();
}

?>
```

Such syntax problem is only revealed at execution time : PHP raises a Fatal error.

The origin of the problem is usually a method that was moved outside a class, at least temporarily.

See also Scope Resolution Operator (::).

Short name	Classes/NoPSSOutsideClass
Rulesets	*Analyze*, *LintButWontExec*
Severity	Major
Time To Fix	Slow (1 hour)

9.774 set_exception_handler() Warning

The set_exception_handler() callable function has to be adapted to PHP 7 : `Exception` is not the right typehint, it is now `Throwable`.

When in doubt about backward compatibility, just drop the typehint. Otherwise, use `Throwable`.

```
<?php

// PHP 5.6- typehint
class foo { function bar(\Exception $e) {} }

// PHP 7+ typehint
```

(continues on next page)

(continued from previous page)

```
class foo { function bar(Throwable $e) {} }

// PHP 5 and PHP 7 compatible typehint (note : there is none)
class foo { function bar($e) {} }

set_exception_handler(foo);

?>
```

Short name	Php/SetExceptionHandlerPHP7
Rulesets	*CompatibilityPHP70*
Severity	Major
Time To Fix	Slow (1 hour)

9.775 strip_tags Skips Closed Tag

strip_tags() skips non-self closing tags. This means that tags such as `<br />` will be ignored from the 2nd argument of the function.

```
<?php

$input = 'a<br />';

// Displays 'a' and clean the tag
echo strip_tags($input, '<br>');

// Displays 'a<br />' and skips the allowed tag
echo strip_tags($input, '<br/>');

?>
```

See also strip_tags.

9.775.1 Suggestions

- Do not use self-closing tags in the 2nd parameter

Short name	Structures/StripTagsSkipsClosedTag
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Quick (30 mins)

9.776 strpos() Too Much

strpos() covers the whole string before reporting 0. If the expected string is expected be at the beginning, or a fixed place, it is more stable to use substr() for comparison.

The longer the haystack (the searched string), the more efficient is that trick. The string has to be 10k or more to have impact, unless it is in a loop.

```
<?php

// This always reads the same amount of string
if (substr($html, 0, 6) === '<html>') {

}

// When searching for a single character, checking with a known position ($string[
↪$position]) is even faster
if ($html[0] === '<') {

}

// With strpos(), the best way is to search for something that exist, and use absence␣
↪as worst case scenario
if (strpos($html, '<html>') > 0) {

} else {
    //
}

// When the search fails, the whole string has been read
if (strpos($html, '<html>') === 0) {

}

?>
```

This applies to stripos() too.

9.776.1 Suggestions

- Check for presence, and not for absence
- use substr() and compare the extracted string
- For single chars, try using the position in the string

Short name	Performances/StrposTooMuch
Rulesets	*Analyze*, *CI-checks*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*WordPress*

9.777 time() Vs strtotime()

time() is actually faster than strtotime() with ‘now’ key string.

```
<?php

// Faster version
$a = time();
```

(continues on next page)

(continued from previous page)

```
// Slower version
$b = strtotime('now');

?>
```

This is a micro-optimisation. Relative gain is real, but small unless the function is used many times.

9.777.1 Suggestions

- Replace strtotime() with time(). Do not change strtotime() with other value than 'now'.

Short name	Performances/timeVsstrtotime
Rulesets	*Performances*
Severity	Minor
Time To Fix	Instant (5 mins)
Examples	*Woocommerce*

9.778 var_dump()... Usage

var_dump(), print_r() or var_export() should not be left in any production code. They are debugging functions.

```
<?php

if ($error) {
    // Debugging usage of var_dump
    // And major security problem
    var_dump($query);

    // This is OK : the $query is logged, and not displayed
    $this->log(print_r($query, true));
}

?>
```

They may be tolerated during development time, but must be removed so as not to have any chance to be run in production.

9.778.1 Suggestions

- Remove usage of var_dump(), print_r(), var_export() without 2nd argument, and other debug functions.
- Push all logging to an external file, instead of the browser.

Short name	Structures/VardumpUsage
Rulesets	*Analyze*, *Security*, *CI-checks*
Severity	Critical
Time To Fix	Instant (5 mins)
ClearPHP	no-debug-code
Examples	*Tine20*, *Piwigo*

CHAPTER 10

Real Code Cases

10.1 Introduction

All the examples in this section are real code, extracted from major PHP applications.

10.2 Examples

10.2.1 Adding Zero

Thelia

Adding Zero, in core/lib/Thelia/Model/Map/ProfileResourceTableMap.php:250.

This return statement is doing quite a lot, including a buried '0 + $offset'. This call is probably an echo to '1 + $offset', which is a little later in the expression.

```
return serialize(array((string) $row[TableMap::TYPE_NUM == $indexType ? 0 + $offset :␣
↪static::translateFieldName('ProfileId', TableMap::TYPE_PHPNAME, $indexType)],␣
↪(string) $row[TableMap::TYPE_NUM == $indexType ? 1 + $offset :␣
↪static::translateFieldName('ResourceId', TableMap::TYPE_PHPNAME, $indexType)]));
```

OpenEMR

Adding Zero, in interface/forms/fee_sheet/new.php:466:534.

$main_provid is filtered as an integer. $main_supid is then filtered twice : one with the sufficent (int) and then, added with 0.

```
if (!$alertmsg && ($_POST['bn_save'] || $_POST['bn_save_close'] || $_POST['bn_save_
↪stay'])) {
    $main_provid = 0 + $_POST['ProviderID'];
    $main_supid  = 0 + (int)$_POST['SupervisorID'];
    //.....
```

10.2.2 Ambiguous Array Index

PrestaShop

Ambiguous Array Index, in src/PrestaShopBundle/Install/Install.php:532.

Null, as a key, is actually the empty string.

```
$list = array(
            'products' => _PS_PROD_IMG_DIR_,
            'categories' => _PS_CAT_IMG_DIR_,
            'manufacturers' => _PS_MANU_IMG_DIR_,
            'suppliers' => _PS_SUPP_IMG_DIR_,
            'stores' => _PS_STORE_IMG_DIR_,
            null => _PS_IMG_DIR_.'l/', // Little trick to copy images in img/l/ path␣
↪with all types
        );
```

Mautic

Ambiguous Array Index, in app/bundles/CoreBundle/Entity/CommonRepository.php:314.

True is turned into 1 (integer), and false is turned into 0 (integer).

```
foreach ($metadata->getAssociationMappings() as $field => $association) {
                    if (in_array($association['type'], [ClassMetadataInfo::ONE_TO_ONE,
↪ ClassMetadataInfo::MANY_TO_ONE])) {
                        $baseCols[true][$entityClass][]  = $association['joinColumns
↪'][0]['name'];
                        $baseCols[false][$entityClass][] = $field;
                    }
                }
```

10.2.3 error_reporting() With Integers

SugarCrm

error_reporting() With Integers, in modules/UpgradeWizard/silentUpgrade_step1.php:436.

This only displays E_ERROR, the highest level of error reporting. It should be checked, as it happens in the 'silentUpgrade' script.

```
ini_set('error_reporting', 1);
```

10.2.4 Eval() Usage

XOOPS

Eval() Usage, in htdocs/modules/system/class/block.php:266.

eval() execute code that was arbitrarily stored in $this, in one of the properties. Then, it is sent to output, but collected before reaching the browser, and put again in $content. May be the echo/ob_get_contents() could have been skipped.

```
ob_start();
                    echo eval($this->getVar('content', 'n'));
                    $content = ob_get_contents();
                    ob_end_clean();
```

Mautic

Eval() Usage, in app/bundles/InstallBundle/Configurator/Step/CheckStep.php:238.

create_function() is actually an eval() in disguise : replace it with a closure for code modernization

```
create_function('$cfgValue', 'return $cfgValue > 100;')
```

10.2.5 Exit() Usage

Traq

Exit() Usage, in src/Controllers/attachments.php:75.

This acts as a view. The final 'exit' is meant to ensure that no other piece of data is emitted, potentially polluting the view. This also prevent any code cleaning to happen.

```
/**
     * View attachment page
     *
     * @param integer $attachment_id
     */
    public function action_view($attachment_id)
    {
        // Don't try to load a view
        $this->render['view'] = false;

        header(Content-type: {$this->attachment->type});
        $content_type = explode('/', $this->attachment->type);

        // Check what type of file we're dealing with.
        if($content_type[0] == 'text' or $content_type[0] == 'image') {
            // If the mime-type is text, we can just display it
            // as plain text. I hate having to download files.
            if ($content_type[0] == 'text') {
                header(Content-type: text/plain);
            }
            header("Content-Disposition: filename=\"{$this->attachment->name}\"");
        }
```

(continues on next page)

(continued from previous page)

```
        // Anything else should be downloaded
        else {
            header("Content-Disposition: attachment; filename=\"{$this->attachment->
↪name}\"");
        }

        // Decode the contents and display it
        print(base64_decode($this->attachment->contents));
        exit;
    }
```

ThinkPHP

Exit() Usage, in ThinkPHP/Library/Vendor/EaseTemplate/template.core.php:60.

Here, exit is used as a rudimentary error management. When the version is not correctly provided via EaseTemplateVer, the application stop totally.

```
$this->version              = (trim($_GET['EaseTemplateVer']))?die('Ease Templae E3!
↪'):'';
```

10.2.6 Multiply By One

SugarCrm

Multiply By One, in SugarCE-Full-6.5.26/modules/Relationships/views/view.editfields.php:74.

Here, '$count % 1' is always true, after the first loop of the foreach. There is no need for % usage.

```
$count = 0;
        foreach($this->fields as $def)
        {
            if (!empty($def['relationship_field'])) {
                $label = !empty($def['vname']) ? $def['vname'] : $def['name'];
                echo <td> . translate($label, $this->module) . :</td>
                   . <td><input id='{$def['name']}' name='{$def['name']}'>  ;

                if ($count%1)
                    echo </tr><tr>;
                $count++;
            }
        }
        echo </tr></table></form>;
```

Edusoho

Multiply By One, in wp-admin/includes/misc.php:74.

1 is useless here, since 24 * 3600 is already an integer. And, of course, a day is not 24 * 3600. . . at least every day.

```
'yesterdayStart' => date('Y-m-d', strtotime(date('Y-m-d', time())) - 1 * 24 * 3600),
```

10.2.7 Not Not

Cleverstyle

Not Not, in modules/OAuth2/OAuth2.php:190.

This double-call returns `$results` as a boolean, preventing a spill of data to the calling method. The `(bool)` operator would be clearer here.

```
$result = $this->db_prime()->q(
                    [
                            DELETE FROM `[prefix]oauth2_clients`
                            WHERE `id` = '%s',
                            DELETE FROM `[prefix]oauth2_clients_grant_access`
                            WHERE `id`      = '%s',
                            DELETE FROM `[prefix]oauth2_clients_sessions`
                            WHERE `id`      = '%s'
                    ],
                    $id
            );
            unset($this->cache->{'/'});
            return !!$result;
```

Tine20

Not Not, in tine20/Calendar/Controller/MSEventFacade.php:392.

It seems that !! is almost superfluous, as a property called 'is_deleted' should already be a boolean.

```
foreach ($exceptions as $exception) {
                $exception->assertAttendee($this->getCalendarUser());
                $this->_prepareException($savedEvent, $exception);
                $this->_preserveMetaData($savedEvent, $exception, true);
                $this->_eventController->createRecurException($exception, !!
↪$exception->is_deleted);
            }
```

10.2.8 include_once() Usage

XOOPS

include_once() Usage, in /htdocs/xoops_lib/modules/protector/admin/center.php:5.

Loading() classes should be down with autoload(). autload() may be build in several distinct functions, using spl_autoload_register().

```
require_once dirname(__DIR__) . 'class/gtickets.php'
```

Tikiwiki

include_once() Usage, in tiki-mytiki_shared.php :140.

Turn the code from tiki-mytiki_shared.php into a function or a method, and call it when needed.

```
include_once('tiki-mytiki_shared.php');
```

10.2.9 Strpos()-like Comparison

Piwigo

Strpos()-like Comparison, in admin/include/functions.php:2585.

preg_match may return 0 if not found, and null if the $pattern is erroneous. While hardcoded regex may be checked at compile time, dynamically built regex may fail at execution time. This is particularly important here, since the function may be called with incoming data for maintenance : 'clear_derivative_cache($_GET['type']);' is in the /admin/maintenance.php.

```
function clear_derivative_cache_rec($path, $pattern)
{
  $rmdir = true;
  $rm_index = false;

  if ($contents = opendir($path))
  {
    while (($node = readdir($contents)) !== false)
    {
      if ($node == '.' or $node == '..')
        continue;
      if (is_dir($path.'/'.$node))
      {
        $rmdir &= clear_derivative_cache_rec($path.'/'.$node, $pattern);
      }
      else
      {
        if (preg_match($pattern, $node))
```

Thelia

Strpos()-like Comparison, in core/lib/Thelia/Controller/Admin/FileController.php:198.

preg_match is used here to identify files with a forbidden extension. The actual list of extension is provided to the method via the parameter $extBlackList, which is an array. In case of mis-configuration by the user of this array, preg_match may fail : for example, when regex special characters are provided. At that point, the whole filter becomes invalid, and can't distinguish good files (returning false) and other files (returning NULL). It is safe to use === false in this situation.

```
if (!empty($extBlackList)) {
            $regex = "#^(.+)\.(".implode("|", $extBlackList).")$#i";

            if (preg_match($regex, $realFileName)) {
                $message = $this->getTranslator()
```

(continues on next page)

(continued from previous page)

```
                    ->trans(
                        'Files with the following extension are not allowed:
↪%extension, please do an archive of the file if you want to upload it',
                        [
                            '%extension' => $fileBeingUploaded->
↪getClientOriginalExtension(),
                        ]
                    );
            }
        }
```

10.2.10 var_dump()… Usage

Tine20

var_dump()… Usage, in tine20/library/Ajam/Connection.php:122.

Two usage of var_dump(). They are protected by configuration, since the debug property must be set to 'true'. Yet, it is safer to avoid them altogether, and log the information to an external file.

```
if($this->debug === true) {
            var_dump($this->getLastRequest());
            var_dump($response);
        }
```

Piwigo

var_dump()… Usage, in include/ws_core.inc.php:273.

This is a hidden debug system : when the response format is not available, the whole object is dumped in the output.

```
function run()
  {
    if ( is_null($this->_responseEncoder) )
    {
      set_status_header(400);
      @header("Content-Type: text/plain");
      echo ("Cannot process your request. Unknown response format.
Request format: ".@$this->_requestFormat." Response format: ".@$this->_responseFormat.
↪"\n");
      var_export($this);
      die(0);
    }
```

10.2.11 Empty Function

Contao

Empty Function, in core-bundle/src/Resources/contao/modules/ModuleQuicklink.php:91.

The closure used with array_map() is empty : this means that the keys are all set to the returned value of the empty closure, which is null. The actual effect is to reset the values to NULL. A better solution, without using the empty closure, is to rely on array_fill_keys() to create an array with default values.

```
if (!empty($tmp) && \is_array($tmp))
                    {
                            $arrPages = array_map(function () {}, array_flip($tmp));
                    }
```

10.2.12 Used Once Variables

shopware

Used Once Variables, in _sql/migrations/438-add-email-template-header-footer-fields.php:115.

In the updateEmailTemplate method, $generatedQueries collects all the generated SQL queries. $generatedQueries is not initialized, and never used after initialization.

```
private function updateEmailTemplate($name, $content, $contentHtml = null)
    {
        $sql = <<<SQL
UPDATE `s_core_config_mails` SET `content` = "$content" WHERE `name` = "$name" AND␣
↪dirty = 0
SQL;
        $this->addSql($sql);

        if ($contentHtml != null) {
            $sql = <<<SQL
UPDATE `s_core_config_mails` SET `content` = "$content", `contentHTML` = "$contentHtml
↪" WHERE `name` = "$name" AND dirty = 0
SQL;
            $generatedQueries[] = $sql;
        }

        $this->addSql($sql);
    }
```

Vanilla

Used Once Variables, in library/core/class.configuration.php:1461.

In this code, $cachedConfigData is collected after storing date in the cache. Gdn::cache()->store() does actual work, so its calling is necessary. The result, collected after execution, is not reused in the rest of the method (long method, not all is shown here). Removing such variable is a needed clean up after development and debug, but also prevents pollution of the variable namespace.

```
// Save to cache if we're into that sort of thing
                $fileKey = sprintf(Gdn_Configuration::CONFIG_FILE_CACHE_KEY, $this->
↪Source);
                if ($this->Configuration && $this->Configuration->caching() &&␣
↪Gdn::cache()->type() == Gdn_Cache::CACHE_TYPE_MEMORY && Gdn::cache()->
↪activeEnabled()) {
                    $cachedConfigData = Gdn::cache()->store($fileKey, $data, [
```

(continues on next page)

(continued from previous page)

```
                        Gdn_Cache::FEATURE_NOPREFIX => true,
                        Gdn_Cache::FEATURE_EXPIRY => 3600
                    ]);
                }
```

10.2.13 Empty Classes

WordPress

Empty Classes, in wp-includes/SimplePie/Core.php:54.

Empty class, but documented as backward compatibility.

```
/**
 * SimplePie class.
 *
 * Class for backward compatibility.
 *
 * @deprecated Use {@see SimplePie} directly
 * @package SimplePie
 * @subpackage API
 */
class SimplePie_Core extends SimplePie
{

}
```

10.2.14 Non Ascii Variables

Magento

Non Ascii Variables, in dev/tests/functional/tests/app/Mage/Checkout/Test/Constraint/AssertOrderWithMultishippingSuccessPlacedMes

The initial C is actually a russian C.

```
$heckoutMultishippingSuccess
```

10.2.15 Non Static Methods Called In A Static

Dolphin

Non Static Methods Called In A Static, in Dolphin-v.7.3.5/xmlrpc/BxDolXMLRPCFriends.php:11.

getIdByNickname() is indeed defined in the class 'BxDolXMLRPCUtil' and it calls the database. The class relies on functions (not methods) to query the database with the correct connexion.

```
class BxDolXMLRPCFriends
{
    function getFriends($sUser, $sPwd, $sNick, $sLang)
    {
        $iIdProfile = BxDolXMLRPCUtil::getIdByNickname ($sNick);
```

Magento

Non Static Methods Called In A Static, in app/code/core/Mage/Paypal/Model/Payflowlink.php:143.

Mage_Payment_Model_Method_Abstract is an abstract class : this way, it is not possible to instantiate it and then, access its methods. The class is extended, so it could be called from one of the objects. Although, the troubling part is that isAvailable() uses $this, so it can't be static.

```
Mage_Payment_Model_Method_Abstract::isAvailable($quote)
```

10.2.16 Forgotten Visibility

FuelCMS

Forgotten Visibility, in /fuel/modules/fuel/controllers/Module.php:713.

Missing visibility for the index() method,and all the methods in the Module class.

```
class Module extends Fuel_base_controller {

    // --------------------------------------------------------------------

    /**
     * Displays the list (table) view
     *
     * @access      public
     * @return      void
     */
    function index()
    {
            $this->items();
    }
```

LiveZilla

Forgotten Visibility, in livezilla/_lib/objects.global.users.inc.php:2516.

Static method that could be public.

```
class Visitor extends BaseUser
{
// Lots of code

    static function CreateSPAMFilter($_userId,$_base64=true)
    {
        if(!empty(Server::$Configuration->File[gl_sfa]))
        {
```

10.2.17 Multiple Index Definition

Magento

Multiple Index Definition, in app/code/core/Mage/Adminhtml/Block/System/Convert/Gui/Grid.php:80.

'type' is defined twice. The first one, 'options' is overwritten.

```
$this->addColumn('store_id', array(
            'header'    => Mage::helper('adminhtml')->__('Store'),
            'type'      => 'options',
            'align'     => 'center',
            'index'     => 'store_id',
            'type'      => 'store',
            'width'     => '200px',
        ));
```

MediaWiki

Multiple Index Definition, in resources/Resources.php:223.

'target' is repeated, though with the same values. This is just dead code.

```
// inside a big array
    'jquery.getAttrs' => [
            'targets' => [ 'desktop', 'mobile' ],
            'scripts' => 'resources/src/jquery/jquery.getAttrs.js',
            'targets' => [ 'desktop', 'mobile' ],
    ],
    // big array continues
```

10.2.18 Incompilable Files

xataface

Incompilable Files, in lib/XML/Tree.php:289.

Compilation error with PHP 7.2 version.

```
syntax error, unexpected 'new' (T_NEW)
```

10.2.19 Multiple Constant Definition

Dolibarr

Multiple Constant Definition, in htdocs/main.inc.php:914.

All is documented here : 'Constants used to defined number of lines in textarea'. Constants are not changing during an execution, and this allows the script to set values early in the process, and have them used later, in the templates. Yet, building constants dynamically may lead to confusion, when developpers are not aware of the change.

```
// Constants used to defined number of lines in textarea
if (empty($conf->browser->firefox))
{
    define('ROWS_1',1);
    define('ROWS_2',2);
    define('ROWS_3',3);
```

(continues on next page)

(continued from previous page)

```
    define('ROWS_4',4);
    define('ROWS_5',5);
    define('ROWS_6',6);
    define('ROWS_7',7);
    define('ROWS_8',8);
    define('ROWS_9',9);
}
else
{
    define('ROWS_1',0);
    define('ROWS_2',1);
    define('ROWS_3',2);
    define('ROWS_4',3);
    define('ROWS_5',4);
    define('ROWS_6',5);
    define('ROWS_7',6);
    define('ROWS_8',7);
    define('ROWS_9',8);
}
```

OpenConf

Multiple Constant Definition, in modules/request.php:71.

The constant is build according to the situation, in the part of the script (file request.php). This hides the actual origin of the value, but keeps the rest of the code simple. Just keep in mind that this constant may have different values.

```
if (isset($_GET['ocparams']) && !empty($_GET['ocparams'])) {
            $params = '';
            if (preg_match_all("/(\w+)--(\w+)_-/", $_GET['ocparams'], $matches)) {
                    foreach ($matches[1] as $idx => $m) {
                            if (($m != 'module') && ($m != 'action') && preg_match("/^
↪[\w-]+$/", $m)) {
                                    $params .= '&' . $m . '=' . urlencode($matches[2][
↪$idx]);
                                    $_GET[$m] = $matches[2][$idx];
                            }
                    }
            }
            unset($_GET['ocparams']);
            define('OCC_SELF', $_SERVER['PHP_SELF'] . '?module=' . $_REQUEST['module
↪'] . '&action=' . $_GET['action'] . $params);
    } elseif (isset($_SERVER['REQUEST_URI']) && strstr($_SERVER['REQUEST_URI'], '?'))
↪{
            define('OCC_SELF', htmlspecialchars($_SERVER['REQUEST_URI']));
    } elseif (isset($_SERVER['QUERY_STRING']) && strstr($_SERVER['QUERY_STRING'], '&
↪')) {
            define('OCC_SELF', $_SERVER['PHP_SELF'] . '?' . htmlspecialchars($_SERVER[
↪'QUERY_STRING']));
    } else {
            err('This server does not support REQUEST_URI or QUERY_STRING','Error');
    }
```

10.2.20 Invalid Constant Name

OpenEMR

Invalid Constant Name, in library/classes/InsuranceCompany.class.php:20.

Either a copy/paste, or a generated definition file : the file contains 25 constants definition. The constant is not found in the rest of the code.

```
define("INS_TYPE_OTHER_NON-FEDERAL_PROGRAMS", 10);
```

10.2.21 Wrong Optional Parameter

FuelCMS

Wrong Optional Parameter, in fuel/modules/fuel/helpers/validator_helper.php:78.

The $regex parameter should really be first, as it is compulsory. Though, if this is a legacy function, it may be better to give regex a default value, such as empty string or null, and test it before using it.

```
if (!function_exists('regex'))
{
    function regex($var = null, $regex)
    {
            return preg_match('#'.$regex.'#', $var);
    }
}
```

Vanilla

Wrong Optional Parameter, in applications/dashboard/modules/class.navmodule.php:99.

Note the second parameter, $dropdown, which has no default value. It is relayed to the addDropdown method, which as no default value too. Since both methods are documented, we can see that they should be an addDropdown : null is probably a good idea, coupled with an explicit check on the actual value.

```
/**
     * Add a dropdown to the items array if it satisfies the $isAllowed condition.
     *
     * @param bool|string|array $isAllowed Either a boolean to indicate whether to␣
↪actually add the item
     * or a permission string or array of permission strings (full match) to check.
     * @param DropdownModule $dropdown The dropdown menu to add.
     * @param string $key The item's key (for sorting and CSS targeting).
     * @param string $cssClass The dropdown wrapper's CSS class.
     * @param array|int $sort Either a numeric sort position or and array in the␣
↪style: array('before|after', 'key').
     * @return NavModule $this The calling object.
     */
    public function addDropdownIf($isAllowed = true, $dropdown, $key = '', $cssClass␣
↪= '', $sort = []) {
        if (!$this->isAllowed($isAllowed)) {
            return $this;
```

(continues on next page)

(continued from previous page)

```
        } else {
            return $this->addDropdown($dropdown, $key, $cssClass, $sort);
        }
    }
```

10.2.22 One Variable String

Tikiwiki

One Variable String, in lib/wiki-plugins/wikiplugin_addtocart.php:228.

Double-quotes are not needed here. If casting to string is important, the (string) would be more explicit.

```
foreach ($plugininfo['params'] as $key => $param) {
            $default["$key"] = $param['default'];
    }
```

NextCloud

One Variable String, in build/integration/features/bootstrap/BasicStructure.php:349.

Both concatenations could be merged, independantly. If readability is important, why not put them inside curly brackets?

```
public static function removeFile($path, $filename) {
            if (file_exists("$path" . "$filename")) {
                    unlink("$path" . "$filename");
            }
    }
```

10.2.23 Static Methods Can't Contain $this

xataface

Static Methods Can't Contain $this, in Dataface/LanguageTool.php:48.

$this is hidden in the arguments of the static call to the method.

```
public static function loadRealm($name){
            return self::getInstance($this->app->_conf['default_language'])->
↪loadRealm($name);
    }
```

SugarCrm

Static Methods Can't Contain $this, in SugarCE-Full-6.5.26/modules/ACLActions/ACLAction.php:332.

Notice how $this is tested for existence before using it. It seems strange, at first, but we have to remember that if $this is never set when calling a static method, a static method may be called with $this. Confusingly, this static method may be called in two ways.

```
static function hasAccess($is_owner=false, $access = 0){

        if($access != 0 && $access == ACL_ALLOW_ALL || ($is_owner && $access == ACL_
↪ALLOW_OWNER))return true;
       //if this exists, then this function is not static, so check the aclaccess␣
↪parameter
        if(isset($this) && isset($this->aclaccess)){
            if($this->aclaccess == ACL_ALLOW_ALL || ($is_owner && $this->aclaccess ==␣
↪ACL_ALLOW_OWNER))
            return true;
        }
        return false;
    }
```

10.2.24 While(List() = Each())

OpenEMR

While(List() = Each()), in library/report.inc:153.

The first while() is needed, to read the arbitrary long list returned by the SQL query. The second list may be upgraded with a foreach, to read both the key and the value. This is certainly faster to execute and to read.

```
function getInsuranceReport($pid, $type = primary)
{
    $sql = select * from insurance_data where pid=? and type=? order by date ASC;
    $res = sqlStatement($sql, array($pid, $type));
    while ($list = sqlFetchArray($res)) {
        while (list($key, $value) = each($list)) {
            if ($ret[$key]['content'] != $value && $ret[$key]['date'] < $list['date
↪']) {
                $ret[$key]['content'] = $value;
                $ret[$key]['date'] = $list['date'];
            }
        }
    }

    return $ret;
}
```

Dolphin

While(List() = Each()), in Dolphin-v.7.3.5/modules/boonex/forum/classes/Forum.php:1875.

This clever use of while() and list() is actually a foreach($a as $r) (the keys are ignored)

```
function getRssUpdatedTopics ()
    {
        global $gConf;

        $this->_rssPrepareConf ();

        $a = $this->fdb->getRecentTopics (0);

        $items = '';
        $lastBuildDate = '';
        $ui = array();
        reset ($a);
        while ( list (,$r) = each ($a) ) {
            // acquire user info
            if (!isset($ui[$r['last_post_user']]) && ($aa = $this->_
↪getUserInfoReadyArray ($r['last_post_user'], false)))
                $ui[$r['last_post_user']] = $aa;

            $td = orca_mb_replace('/#/', $r['count_posts'], '[L[# posts]]') . ' &#183;
↪ ' . orca_mb_replace('/#/', $ui[$r['last_post_user']]['title'], '[L[last reply by
↪#]]') . ' &#183; ' . $r['cat_name'] . ' &#187; ' . $r['forum_title'];
```

10.2.25 Several Instructions On The Same Line

Piwigo

Several Instructions On The Same Line, in tools/triggers_list.php:993.

There are two instructions on the line with the if(). Note that the condition is not followed by a bracketed block. When reviewing, it really seems that echo '<br>' and $f=0; are on the same block, but the second is indeed an unconditional expression. This is very difficult to spot.

```
foreach ($trigger['files'] as $file)
      {
        if (!$f) echo '<br>'; $f=0;
        echo preg_replace('#\((.+)\)#', '(<i>$1</i>)', $file);
      }
```

Tine20

Several Instructions On The Same Line, in tine20/Calendar/Controller/Event.php:1594.

Here, $_event->attendee is saved in a local variable, then the property is destroyed. Same for $_event->notes; Strangely, a few lines above, the properties are unset on their own line. Unsetting properties leads to surprise bugs, and hidding the unset after ; makes it harder to spot.

```
$futurePersistentExceptionEvents->setRecurId($_event->getId());
                unset($_event->recurid);
                unset($_event->base_event_id);
                foreach(array('attendee', 'notes', 'alarms') as $prop) {
                    if ($_event->{$prop} instanceof Tinebase_Record_RecordSet) {
                        $_event->{$prop}->setId(NULL);
```

(continues on next page)

(continued from previous page)

```
                    }
                }
                $_event->exdate = $futureExdates;

                $attendees = $_event->attendee; unset($_event->attendee);
                $note = $_event->notes; unset($_event->notes);
                $persistentExceptionEvent = $this->create($_event, $_
↪checkBusyConflicts && $dtStartHasDiff);
```

10.2.26 Multiples Identical Case

SugarCrm

Multiples Identical Case, in modules/ModuleBuilder/MB/MBPackage.php:439.

It takes a while to find the double 'required' case, but the executed code is actually the same, so this is dead code at worst.

```
switch ($col) {
    case 'custom_module':
            $installdefs['custom_fields'][$name]['module'] = $res;
            break;
    case 'required':
            $installdefs['custom_fields'][$name]['require_option'] = $res;
            break;
    case 'vname':
            $installdefs['custom_fields'][$name]['label'] = $res;
            break;
    case 'required':
            $installdefs['custom_fields'][$name]['require_option'] = $res;
            break;
    case 'massupdate':
            $installdefs['custom_fields'][$name]['mass_update'] = $res;
            break;
    case 'comments':
            $installdefs['custom_fields'][$name]['comments'] = $res;
            break;
    case 'help':
            $installdefs['custom_fields'][$name]['help'] = $res;
            break;
    case 'len':
            $installdefs['custom_fields'][$name]['max_size'] = $res;
            break;
    default:
            $installdefs['custom_fields'][$name][$col] = $res;
}//switch
```

ExpressionEngine

Multiples Identical Case, in ExpressionEngine_Core2.9.2/system/expressionengine/controllers/cp/admin_content.php:577.

'deft_status' is doubled, with a fallthrough. This looks like some forgotten copy/paste.

```
switch ($key){
                                                            case 'cat_group':
                                                                //PHP code
                                                                    break;
                                                            case 'status_group':
                                                            case 'field_group':
                                                                //PHP code
                                                                    break;
                                                            case 'deft_status':
                                                            case 'deft_status':
                                                                //PHP code
                                                                    break;
                                                            case 'search_excerpt':
                                                                //PHP code
                                                                    break;
                                                            case 'deft_category':
                                                                //PHP code
                                                                    break;
                                                            case 'blog_url':
                                                            case 'comment_url':
                                                            case 'search_results_url':
                                                            case 'rss_url':
                                                                //PHP code
                                                                    break;
                                                            default :
                                                                //PHP code
                                                                    break;
                                                    }
```

10.2.27 Switch Without Default

Zencart

Switch Without Default, in admin/tax_rates.php:15.

The ‘action’ is collected from $_GET and then, compared with various strings to handle the different actions to be taken. The default behavior is implicit here : if no ‘action’, display the initial form for taxes to be changed. This has to be understood as a general philosophy of ZenCart project, or by reading the rest of the HTML code. Adding a ‘default’ case here would help understand what happens in case ‘action’ is absent or unrecognized.

```
$action = (isset($_GET['action']) ? $_GET['action'] : '');

  if (zen_not_null($action)) {
    switch ($action) {
      case 'insert':
        // PHP code
        break;
      case 'save':
        // PHP code
        break;
      case 'deleteconfirm':
        // PHP code
        break;
    }
  }
?> .... HTML code
```

Traq

Switch Without Default, in src/Helpers/Ticketlist.php:311.

The default case is actually processed after the switch, by the next if/then structure. The structure deals with the customFields, while the else deals with any unknown situations. This if/then could be wrapped in the 'default' case of switch, for consistent processing. The if/then condition would be hard to use as a 'case' (possible, though).

```
public static function dataFor($column, $ticket)
    {
        switch ($column) {
            // Ticket ID column
            case 'ticket_id':
                return $ticket['ticket_id'];
                break;

            // Status column
            case 'status':
            case 'type':
            case 'component':
            case 'priority':
            case 'severity':
                return $ticket[{$column}_name];
                break;

            // Votes
            case 'votes':
                return $ticket['votes'];
                break;
        }

        // If we're still here, it may be a custom field
        if ($value = $ticket->customFieldValue($column)) {
            return $value->value;
        }

        // Nothing!
        return '';
    }
```

10.2.28 $this Belongs To Classes Or Traits

OpenEMR

$this Belongs To Classes Or Traits, in ccr/display.php:24.

$this is used to call the document_upload_download_log() method, although this piece of code is not part of a class, nor is included in a class.

```
<?php
require_once(dirname(__FILE__) . "/../interface/globals.php");

$type = $_GET['type'];
```

(continues on next page)

(continued from previous page)

```
$document_id = $_GET['doc_id'];
$d = new Document($document_id);
$url =  $d->get_url();
$storagemethod = $d->get_storagemethod();
$couch_docid = $d->get_couch_docid();
$couch_revid = $d->get_couch_revid();

if ($couch_docid && $couch_revid) {
    $couch = new CouchDB();
    $data = array($GLOBALS['couchdb_dbase'],$couch_docid);
    $resp = $couch->retrieve_doc($data);
    $xml = base64_decode($resp->data);
    if ($content=='' && $GLOBALS['couchdb_log']==1) {
        $log_content = date('Y-m-d H:i:s')." ==> Retrieving document\r\n";
        $log_content = date('Y-m-d H:i:s')." ==> URL: ".$url."\r\n";
        $log_content .= date('Y-m-d H:i:s')." ==> CouchDB Document Id: ".$couch_docid.
↪"\r\n";
        $log_content .= date('Y-m-d H:i:s')." ==> CouchDB Revision Id: ".$couch_revid.
↪"\r\n";
        $log_content .= date('Y-m-d H:i:s')." ==> Failed to fetch document content␣
↪from CouchDB.\r\n";
        //$log_content .= date('Y-m-d H:i:s')." ==> Will try to download file from␣
↪HardDisk if exists.\r\n\r\n";
        $this->document_upload_download_log($d->get_foreign_id(), $log_content);
        die(xlt("File retrieval from CouchDB failed"));
    }
```

10.2.29 Nested Ternary

SPIP

Nested Ternary, in ecrire/inc/utils.php:2648.

Interesting usage of both if/then, for the flow control, and ternary, for data process. Even on multiple lines, nested ternaries are quite hard to read.

```
// le script de l'espace prive
    // Mettre a "index.php" si DirectoryIndex ne le fait pas ou pb connexes:
    // les anciens IIS n'acceptent pas les POST sur ecrire/ (#419)
    // meme pb sur thttpd cf. http://forum.spip.net/fr_184153.html
    if (!defined('_SPIP_ECRIRE_SCRIPT')) {
            define('_SPIP_ECRIRE_SCRIPT', (empty($_SERVER['SERVER_SOFTWARE']) ? '' :
                    preg_match(',IIS|thttpd,', $_SERVER['SERVER_SOFTWARE']) ?
                            'index.php' : ''));
    }
```

Zencart

Nested Ternary, in app/library/zencart/ListingQueryAndOutput/src/formatters/TabularProduct.php:143.

No more than one level of nesting for this ternary call, yet it feels a lot more, thanks to the usage of arrayed properties, constants, and functioncalls.

```
$lc_text .= '<br />' . (zen_get_show_product_switch($listing->fields['products_id'],
↪'ALWAYS_FREE_SHIPPING_IMAGE_SWITCH') ? (zen_get_product_is_always_free_shipping(
↪$listing->fields['products_id']) ? TEXT_PRODUCT_FREE_SHIPPING_ICON . '<br />' : '')␣
↪: '');
```

10.2.30 Non-constant Index In Array

Dolibarr

Non-constant Index In Array, in htdocs/includes/OAuth/Common/Storage/DoliStorage.php:245.

The *state* constant in the *$result* array is coming from the SQL query. There is no need to make this a constant : making it a string will remove some warnings in the logs.

```
public function hasAuthorizationState($service)
    {
        // get state from db
        dol_syslog("get state from db");
        $sql = "SELECT state FROM ".MAIN_DB_PREFIX."oauth_state";
        $sql.= " WHERE service='".$this->db->escape($service)."'";
        $resql = $this->db->query($sql);
        $result = $this->db->fetch_array($resql);
        $states[$service] = $result[state];
        $this->states[$service] = $states[$service];

        return is_array($states)
        && isset($states[$service])
        && null !== $states[$service];
    }
```

Zencart

Non-constant Index In Array, in app/library/zencart/Services/src/LeadLanguagesRoutes.php:112.

The *fields* constant in the *$tableEntry* which holds a list of tables. It seems to be a SQL result, but it is conveniently abstracted with *$this->listener->getTableList()*, so we can't be sure.

```
public function updateLanguageTables($insertId)
    {
        $tableList = $this->listener->getTableList();
        if (count($tableList) == 0) {
            return;
        }
        foreach ($tableList as $tableEntry) {
            $languageKeyField = issetorArray($tableEntry, 'languageKeyField',
↪'language_id');
            $sql = " INSERT IGNORE INTO :table: (";
            $sql = $this->dbConn->bindVars($sql, ':table:', $tableEntry ['table'],
↪'noquotestring');
            $sql .= $languageKeyField. ", ";
            $fieldNames = "";
            foreach ($tableEntry[fields] as $fieldName => $fieldType) {
```

(continues on next page)

(continued from previous page)

```
                $fieldNames .= $fieldName . ", ";
            }
```

10.2.31 Class, Interface Or Trait With Identical Names

shopware

Class, Interface Or Trait With Identical Names, in engine/Shopware/Components/Form/Interfaces/Element.php:30.

Most Element classes extends ModelEntity, which is an abstract class. There is also an interface, called Element, for forms. And, last, one of the class Element extends JsonSerializable, which is a PHP native interface. Namespaces are definitely crucial to understand which Element is which.

```
interface Element { /**/ } // in engine/Shopware/Components/Form/Interfaces/Element.
↪php:30

class Element implements \JsonSerializable { /**/ }         // in engine/Shopware/
↪Bundle/EmotionBundle/Struct/Element.php:29

class Element extends ModelEntity { /**/ }  // in /engine/Shopware/Models/Document/
↪Element.php:37
```

NextCloud

Class, Interface Or Trait With Identical Names, in lib/private/Files/Storage/Storage.php:33.

Interface Storage extends another Storage class. Here, the fully qualified name is used, so we can understand which storage is which at read time : a 'use' alias would make this line more confusing.

```
interface Storage extends \OCP\Files\Storage { /**/ }
```

10.2.32 Empty Try Catch

LiveZilla

Empty Try Catch, in livezilla/_lib/trdp/Zend/Mail/Protocol/Pop3.php:237.

This is an aptly commented empty try/catch : the emited exception is extra check for a Zend Mail Protocol Exception. Hopefully, the Zend_Mail_Protocol_Exception only covers a already-closed situation. Anyhow, this should be logged for later diagnostic.

```
public function logout()
    {
        if (!$this->_socket) {
            return;
        }

        try {
            $this->request('QUIT');
        } catch (Zend_Mail_Protocol_Exception $e) {
```

(continues on next page)

(continued from previous page)

```
            // ignore error - we're closing the socket anyway
        }

        fclose($this->_socket);
        $this->_socket = null;
    }
```

Mautic

Empty Try Catch, in app/bundles/ReportBundle/Model/ExportHandler.php:66.

Removing a file : if the file is not 'deleted' by the method call, but raises an error, it is hidden. When file destruction is impossible because the file is already destroyed (or missing), this is well. If the file couldn't be destroyed because of missing writing privileges, hiding this error will have serious consequences.

```
/**
     * @param string $fileName
     */
    public function removeFile($fileName)
    {
        try {
            $path = $this->getPath($fileName);
            $this->filePathResolver->delete($path);
        } catch (FileIOException $e) {
        }
    }
```

10.2.33 Used Once Variables (In Scope)

shopware

Used Once Variables (In Scope), in _sql/migrations/438-add-email-template-header-footer-fields.php:115.

In the updateEmailTemplate method, $generatedQueries collects all the generated SQL queries. $generatedQueries is not initialized, and never used after initialization.

```
private function updateEmailTemplate($name, $content, $contentHtml = null)
    {
        $sql = <<<SQL
UPDATE `s_core_config_mails` SET `content` = "$content" WHERE `name` = "$name" AND␣
↪dirty = 0
SQL;
        $this->addSql($sql);

        if ($contentHtml != null) {
            $sql = <<<SQL
UPDATE `s_core_config_mails` SET `content` = "$content", `contentHTML` = "$contentHtml
↪" WHERE `name` = "$name" AND dirty = 0
SQL;
            $generatedQueries[] = $sql;
        }

```

(continues on next page)

(continued from previous page)

```
        $this->addSql($sql);
    }
```

10.2.34 Deprecated Functions

Dolphin

Deprecated Functions, in Dolphin-v.7.3.5/inc/classes/BxDolAdminSettings.php:270.

Split() was abandonned in PHP 7.0

```
split(',', $aItem['extra']);
```

10.2.35 Dangling Array References

Typo3

Dangling Array References, in typo3/sysext/impexp/Classes/ImportExport.php:322.

foreach() reads $lines into $r, and augment those lines. By the end, the $r variable is not unset. Yet, several lines later, in the same method but with different conditions, another loop reuse the variable $r. If is_array($this->dat['header']['pagetree'] and is_array($this->remainHeader['records']) are arrays at the same moment, then both loops are called, and they share the same reference. Values of the latter array will end up in the formar.

```
if (is_array($this->dat['header']['pagetree'])) {
    reset($this->dat['header']['pagetree']);
    $lines = [];
    $this->traversePageTree($this->dat['header']['pagetree'], $lines);

    $viewData['dat'] = $this->dat;
    $viewData['update'] = $this->update;
    $viewData['showDiff'] = $this->showDiff;
    if (!empty($lines)) {
        foreach ($lines as &$r) {
            $r['controls'] = $this->renderControls($r);
            $r['fileSize'] = GeneralUtility::formatSize($r['size']);
            $r['message'] = ($r['msg'] && !$this->doesImport ? '<span class=text-
↪danger>' . htmlspecialchars($r['msg']) . '</span>' : '');
        }
        $viewData['pagetreeLines'] = $lines;
    } else {
        $viewData['pagetreeLines'] = [];
    }
}
// Print remaining records that were not contained inside the page tree:
if (is_array($this->remainHeader['records'])) {
    $lines = [];
    if (is_array($this->remainHeader['records']['pages'])) {
        $this->traversePageRecords($this->remainHeader['records']['pages'], $lines);
    }
    $this->traverseAllRecords($this->remainHeader['records'], $lines);
    if (!empty($lines)) {
        foreach ($lines as &$r) {
```

(continues on next page)

(continued from previous page)

```
            $r['controls'] = $this->renderControls($r);
            $r['fileSize'] = GeneralUtility::formatSize($r['size']);
            $r['message'] = ($r['msg'] && !$this->doesImport ? '<span class=text-
↪danger>' . htmlspecialchars($r['msg']) . '</span>' : '');
        }
        $viewData['remainingRecords'] = $lines;
    }
}
```

SugarCrm

Dangling Array References, in SugarCE-Full-6.5.26/modules/Import/CsvAutoDetect.php:165.

There are two nested foreach here : they both have referenced blind variables. The second one uses $data, but never changes it. Yet, it is reused the next round in the first loop, leading to pollution from the first rows of $this->_parser->data into the lasts. This may happen even if $data is not modified explicitely : in fact, it will be modified the next call to foreach($row as . . .), for each element in $row.

```
foreach ($this->_parser->data as &$row) {
    foreach ($row as &$data) {
        $len = strlen($data);
        // check if it begins and ends with single quotes
        // if it does, then it double quotes may not be the enclosure
        if ($len>=2 && $data[0] == " && $data[$len-1] == ") {
            $beginEndWithSingle = true;
            break;
        }
    }
    if ($beginEndWithSingle) {
        break;
    }
    $depth++;
    if ($depth > $this->_max_depth) {
        break;
    }
}
```

10.2.36 Queries In Loops

TeamPass

Queries In Loops, in install/install.queries.php:551.

The value is SELECTed first in the database, and it is INSERTed if not. This may be done in one call in most databases.

```
foreach ($aMiscVal as $elem) {
    //Check if exists before inserting
    $tmp = mysqli_num_rows(
        mysqli_query(
            $dbTmp,
            SELECT * FROM `.$var['tbl_prefix'].misc`
            WHERE type='.$elem[0].' AND intitule='.$elem[1].'
```

(continues on next page)

(continued from previous page)

```
        )
    );
    if (intval($tmp) === 0) {
        $queryRes = mysqli_query(
            $dbTmp,
            INSERT INTO `.$var['tbl_prefix'].misc`
            (`type`, `intitule`, `valeur`) VALUES
            ('.$elem[0].', '.$elem[1].', '.
            str_replace(', , $elem[2]).');
        ); // or die(mysqli_error($dbTmp))
    }

    // append new setting in config file
    $config_text .= '.$elem[1].' => '.str_replace(', , $elem[2]).',;
                        }
```

OpenEMR

Queries In Loops, in contrib/util/deidentification/deidentification.php:287.

The value is SELECTed first in the database, and it is INSERTed if not. This may be done in one call in most databases.

```
$query = select * from facility;
$result = mysqli_query($con, $query);
while ($row = mysqli_fetch_array($result)) {
    $string = update facility set

          `name`    = 'Facility_{$row['id']}',
          `phone`   = '(000) 000-0000'

        where `id` = {$row['id']};

    mysqli_query($con, $string) or print Error altering facility table \n;
    $string = '';
}
```

10.2.37 Aliases Usage

Cleverstyle

Aliases Usage, in modules/HybridAuth/Hybrid/thirdparty/Vimeo/Vimeo.php:422.

is_writeable() should be written is_writable(). No extra 'e'.

```
is_writeable($chunk_temp_dir)
```

phpMyAdmin

Aliases Usage, in libraries/classes/Server/Privileges.php:5064.

join() should be written implode()

```
join('`, `', $tmp_privs2['Update'])
```

10.2.38 Var Keyword

xataface

Var Keyword, in SQL/Parser/wrapper.php:24.

With the usage of var and a first method bearing the name of the class, this is PHP 4 code that is still in use.

```
class SQL_Parser_wrapper {

    var $_data;
    var $_tableLookup;
    var $_parser;

    function SQL_Parser_wrapper(&$data, $dialect='MySQL'){
```

10.2.39 Wrong Number Of Arguments

xataface

Wrong Number Of Arguments, in actions/existing_related_record.php:130.

df_display() actually requires only 2 arguments, while three are provided. The last argument is completely ignored. df_display() is called in a total of 9 places : this now looks like an API change that left many calls untouched.

```
df_display($context, $template, true);

// in public-api.php :
function df_display($context, $template_name){
    import( 'Dataface/SkinTool.php');
    $st = Dataface_SkinTool::getInstance();

    return $st->display($context, $template_name);
}
```

10.2.40 Undefined static:: Or self::

xataface

Undefined static:: Or self::, in actions/forgot_password.php:194.

This is probably a typo, since the property called public static $EX_NO_USERS_WITH_EMAIL = 501; is defined in that class.

```
if ( !$user ) throw new Exception(df_translate('actions.forgot_password.null_user',
↪"Cannot send email for null user"), self::$EX_NO_USERS_FOUND_WITH_EMAIL);
```

SugarCrm

Undefined static:: Or self::, in code/SugarCE-Full-6.5.26/include/SugarDateTime.php:574.

self::$sugar_strptime_long_mon refers to the current class, which extends DateTime. No static property was defined at either of them, with the name '$sugar_strptime_long_mon'. This has been a Fatal error at execution time since PHP 5.3, at least.

```
if ( isset($regexp['positions']['F']) && !empty($dateparts[$regexp['positions']['F
↪']])) {
               // FIXME: locale?
    $mon = $dateparts[$regexp['positions']['F']];
    if(isset(self::$sugar_strptime_long_mon[$mon])) {
        $data["tm_mon"] = self::$sugar_strptime_long_mon[$mon];
    } else {
        return false;
    }
}
```

10.2.41 list() May Omit Variables

OpenConf

list() May Omit Variables, in openconf/author/privacy.php:29.

The first variable in the list(), $none, isn't reused anywhere in the script. In fact, its name convey the meaning that is it useless, but is in the array nonetheless.

```
list($none, $OC_privacy_policy) = oc_getTemplate('privacy_policy');
```

FuelCMS

list() May Omit Variables, in wp-admin/includes/misc.php:74.

$a is never reused again. $b, on the other hand is. Not assigning any value to $a saves some memory, and avoid polluting the local variable space.

```
list($b, $a) = array(reset($params->me), key($params->me));
```

10.2.42 Or Die

Tine20

Or Die, in scripts/addgrant.php:34.

Typical error handling, which also displays the MySQL error message, and leaks informations about the system. One may also note that mysql_connect is not supported anymore, and was replaced with mysqli and pdo : this may be a backward compatibile file.

```
$link = mysql_connect($host, $user, $pass) or die("No connection: " . mysql_error( ))
```

OpenConf

Or Die, in openconf/chair/export.inc:143.

or die() is also applied to many situations, where a blocking situation arise. Here, with the creation of a temporary file.

```
$coreFile = tempnam('/tmp/', 'ocexport') or die('could not generate Excel file (6)')
```

10.2.43 Use const

phpMyAdmin

Use const, in error_report.php:17.

This may be turned into a *const* call, with a static expression.

```
define('ROOT_PATH', __DIR__ . DIRECTORY_SEPARATOR)
```

Piwigo

Use const, in include/functions_plugins.inc.php:32.

Const works efficiently with literal

```
define('EVENT_HANDLER_PRIORITY_NEUTRAL', 50)
```

10.2.44 Written Only Variables

Dolibarr

Written Only Variables, in htdocs/ecm/class/ecmdirectory.class.php:692.

$val is only written, as only the keys are used. $val may be skipped by applying the foreach to array_keys($this->cats), instead of the whole array.

```
// We add properties fullxxx to all elements
            foreach($this->cats as $key => $val)
            {
                    if (isset($motherof[$key])) continue;
                    $this->build_path_from_id_categ($key, 0);
            }
```

SuiteCrm

Written Only Variables, in modules/Campaigns/utils.php:820.

$email_health is used later in the method; while $email_components is only set, and never used.

```
//run query for mail boxes of type 'bounce'
        $email_health = 0;
        $email_components = 2;
        $mbox_qry = "select * from inbound_email where deleted ='0' and mailbox_type␣
↪= 'bounce'";
        $mbox_res = $focus->db->query($mbox_qry);

        $mbox = array();
        while ($mbox_row = $focus->db->fetchByAssoc($mbox_res)) {
            $mbox[] = $mbox_row;
        }
```

10.2.45 Foreach Reference Is Not Modified

Dolibarr

Foreach Reference Is Not Modified, in htdocs/product/reassort.php:364.

$wh is an array, and is read for its index 'id', but it is not modified. The reference sign is too much.

```
if($nb_warehouse>1) {
    foreach($warehouses_list as &$wh) {

        print '<td class=right>';
        print empty($product->stock_warehouse[$wh['id']]->real) ? '0' : $product->
↪stock_warehouse[$wh['id']]->real;
        print '</td>';
    }
}
```

Vanilla

Foreach Reference Is Not Modified, in applications/vanilla/models/class.discussionmodel.php:944.

$discussion is also an object : it doesn't need any reference to be modified. And, it is not modified, but only read.

```
foreach ($result as $key => &$discussion) {
    if (isset($this->_AnnouncementIDs)) {
        if (in_array($discussion->DiscussionID, $this->_AnnouncementIDs)) {
            unset($result[$key]);
            $unset = true;
        }
    } elseif ($discussion->Announce && $discussion->Dismissed == 0) {
        // Unset discussions that are announced and not dismissed
        unset($result[$key]);
        $unset = true;
    }
}
```

10.2.46 Useless Return

ThinkPHP

Useless Return, in library/think/Request.php:2121.

__set() doesn't need a return, unlike __get().

```
public function __set($name, $value)
    {
        return $this->param[$name] = $value;
    }
```

Vanilla

Useless Return, in applications/dashboard/views/attachments/attachment.php:14.

The final 'return' is useless : return void (here, return without argument), is the same as returning null, unless the 'void' return type is used. The other return, is in the two conditions, is important to skip the end of the functioncall.

```
function writeAttachment($attachment) {

        $customMethod = AttachmentModel::getWriteAttachmentMethodName($attachment[
↪'Type']);
        if (function_exists($customMethod)) {
            if (val('Error', $attachment)) {
                writeErrorAttachment($attachment);
                return;
            }
            $customMethod($attachment);
        } else {
            trace($customMethod, 'Write Attachment method not found');
            trace($attachment, 'Attachment');
        }
        return;
    }
```

10.2.47 Unpreprocessed Values

Zurmo

Unpreprocessed Values, in app/protected/core/utils/ZurmoTranslationServerUtil.php:79.

It seems that a simple concatenation could be used here. There is another call to this expression in the code, and a third that uses 'PATCH_VERSION' on top of the two others.

```
join('.', array(MAJOR_VERSION, MINOR_VERSION))
```

Piwigo

Unpreprocessed Values, in include/random_compat/random.php:34.

PHP_VERSION is actually build with PHP_MAJOR_VERSION, PHP_MINOR_VERSION and PHP_RELEASE_VERSION. There is also a compact version : PHP_VERSION_ID

```
explode('.', PHP_VERSION);
```

10.2.48 Undefined Properties

WordPress

Undefined Properties, in wp-admin/includes/misc.php:74.

Properties are not defined, but they are thoroughly initialized when the XML document is parsed. All those definition should be in a property definition, for clear documentation.

```
$this->DeliveryLine1 = '';
        $this->DeliveryLine2 = '';
        $this->City = '';
        $this->State = '';
        $this->ZipAddon = '';
```

MediaWiki

Undefined Properties, in includes/logging/LogFormatter.php:561.

parsedParametersDeleteLog is an undefined property. Defining the property with a null default value is important here, to keep the code running.

```
protected function getMessageParameters() {
            if ( isset( $this->parsedParametersDeleteLog ) ) {
                    return $this->parsedParametersDeleteLog;
            }
```

10.2.49 Strict Comparison With Booleans

Phinx

Strict Comparison With Booleans, in src/Phinx/Db/Adapter/MysqlAdapter.php:1131.

ìs*Null( )*' always returns a boolean : it may be only be `true` or `false`. Until typehinted properties or return typehint are used, isNull() may return anything else.

```
$column->isNull( ) == false
```

Typo3

Strict Comparison With Booleans, in typo3/sysext/lowlevel/Classes/Command/FilesWithMultipleReferencesCommand.php:90.

When `dry-run` is not defined, the getOption() method actually returns a `null` value. So, comparing the result of getOption() to false is actually wrong : using a constant to prevent values to be inconsistent is recommended here.

```
$input->getOption('dry-run') != false
```

10.2.50 Lone Blocks

ThinkPHP

Lone Blocks, in ThinkPHP/Library/Vendor/Hprose/HproseReader.php:163.

There is no need for block in a case/default clause. PHP executes all command in order, until a break or the end of the switch. There is another occurrence of that situation in this code : it seems to be a coding convention, while only applied to a few switch statements.

```
for ($i = 0; $i < $len; ++$i) {
            switch (ord($this->stream->getc()) >> 4) {
                case 0:
                case 1:
                case 2:
                case 3:
                case 4:
                case 5:
                case 6:
                case 7: {
                    // 0xxx xxxx
                    $utf8len++;
                    break;
                }
                case 12:
                case 13: {
                    // 110x xxxx   10xx xxxx
                    $this->stream->skip(1);
                    $utf8len += 2;
                    break;
                }
```

Tine20

Lone Blocks, in tine20/Addressbook/Convert/Contact/VCard/Abstract.php:199.

A case of empty case, with empty blocks. This is useless code. Event the curly brackets with the final case are useless.

```
switch ( $property['TYPE'] ) {
                        case 'JPG' : {}
                        case 'jpg' : {}
                        case 'Jpg' : {}
                        case 'Jpeg' : {}
                        case 'jpeg' : {}
```

(continues on next page)

(continued from previous page)

```
                        case 'PNG' : {}
                        case 'png' : {}
                        case 'JPEG' : {
                            if (Tinebase_Core::isLogLevel(Zend_Log::DEBUG))
                                Tinebase_Core::getLogger()->warn(__METHOD__ . '::' . _
↪_LINE__ . ' Photo: passing on invalid ' . $property['TYPE'] . ' image as is (' .␣
↪strlen($property->getValue()) .')' );
                            $jpegphoto = $property->getValue();
                            break;
                        }
```

10.2.51 PHP Keywords As Names

ChurchCRM

PHP Keywords As Names, in src/kiosk/index.php:42.

$false may be true or false (or else…). In fact, the variable is not even defined in this file, and the file do a lot of inclusion.

```
if (!isset($_COOKIE['kioskCookie'])) {
    if ($windowOpen) {
        $guid = uniqid();
        setcookie("kioskCookie", $guid, 2147483647);
        $Kiosk = new \ChurchCRM\KioskDevice();
        $Kiosk->setGUIDHash(hash('sha256', $guid));
        $Kiosk->setAccepted($false);
        $Kiosk->save();
    } else {
        header("HTTP/1.1 401 Unauthorized");
        exit;
    }
}
```

---

xataface

PHP Keywords As Names, in Dataface/Record.php:1278.

This one is documented, and in the end, makes a lot of sense.

```
function &getRelatedRecord($relationshipName, $index=0, $where=0, $sort=0){
            if ( isset($this->cache[__FUNCTION__][$relationshipName][$index][$where][
↪$sort]) ){
                    return $this->cache[__FUNCTION__][$relationshipName][$index][
↪$where][$sort];
            }
            $it = $this->getRelationshipIterator($relationshipName, $index, 1, $where,
↪ $sort);
            if ( $it->hasNext() ){
                    $rec =& $it->next();
                    $this->cache[__FUNCTION__][$relationshipName][$index][$where][
↪$sort] =& $rec;
```

(continues on next page)

(continued from previous page)

```
                    return $rec;
            } else {
                    $null = null;   // stupid hack because literal 'null' can't be␣
↪returned by ref.
                    return $null;
            }
    }
```

10.2.52 Could Use self

WordPress

Could Use self, in wp-admin/includes/misc.php:74.

Securimage could be called self.

```
class Securimage
{
// Lots of code
            Securimage::$_captchaId = $id;
}
```

LiveZilla

Could Use self, in livezilla/_lib/objects.global.users.inc.php:1599.

Using self makes it obvious that Operator::GetSystemId() is a local call, while Communication::GetParameter() is external.

```
class Operator extends BaseUser
{
    static function ReadParams()
    {
        if(!empty($_POST[POST_EXTERN_REQUESTED_INTERNID]))
            return Communication::GetParameter(POST_EXTERN_REQUESTED_INTERNID,,$c,
↪FILTER_SANITIZE_SPECIAL_CHARS,null,32);
        else if(!empty($_GET[operator]))
        {
            $userid = Communication::GetParameter(operator,,$c,FILTER_SANITIZE_
↪SPECIAL_CHARS,null,32,false,false);
            $sysid = Operator::GetSystemId($userid);
}
```

10.2.53 Logical Should Use Symbolic Operators

Cleverstyle

Logical Should Use Symbolic Operators, in modules/Uploader/Mime/Mime.php:171.

$extension is assigned with the results of pathinfo($reference_name, PATHINFO_EXTENSION) and ignores static::hasExtension($extension). The same expression, placed in a condition (like an if), would assign a value to $extension and use another for the condition itself. Here, this code is only an expression in the flow.

```
$extension = pathinfo($reference_name, PATHINFO_EXTENSION) and static::hasExtension(
↪$extension);
```

OpenConf

Logical Should Use Symbolic Operators, in chair/export.inc:143.

In this context, the priority of execution is used on purpose; $coreFile only collect the temporary name of the export file, and when this name is empty, then the second operand of OR is executed, though never collected. Since this second argument is a 'die', its return value is lost, but the initial assignation is never used anyway.

```
$coreFile = tempnam('/tmp/', 'ocexport') or die('could not generate Excel file (6)')
```

10.2.54 Catch Overwrite Variable

PhpIPAM

Catch Overwrite Variable, in app/subnets/scan/subnet-scan-snmp-route.php:58.

$e is used both as 'local' variable : it is local to the catch clause, and it is a blind variable in a foreach(). There is little overlap between the two occurrences, but one reader may wonder why the caught exception is shown later on.

```
try {
        $res = $Snmp->get_query(get_routing_table);
        // remove those not in subnet
        if (sizeof($res)>0) {
           // save for debug
           $debug[$d->hostname][$q] = $res;

           // save result
           $found[$d->id][$q] = $res;
        }
    } catch (Exception $e) {
       // save for debug
       $debug[$d->hostname][$q] = $res;
       $errors[] = $e->getMessage();
    }

// lots of code
// on line 132
    // print errors
    if (isset($errors)) {
        print <hr>;
        foreach ($errors as $e) {
            print $Result->show (warning, $e, false, false, true);
        }
    }
```

SuiteCrm

Catch Overwrite Variable, in modules/Emails/EmailUIAjax.php:1082.

$e starts as an Email(), in the 'getMultipleMessagesFromSugar' case, while a few lines later, in 'refreshSugarFolders', $e is now an exception. Breaks are in place, so both occurrences are separated, yet, one may wonder why an email is a warning, or a mail is a warning.

```
// On line 900, $e is a Email
        case getMultipleMessagesFromSugar:
            $GLOBALS['log']->debug(********** EMAIL 2.0 - Asynchronous - at:␣
↪getMultipleMessagesFromSugar);
            if (isset($_REQUEST['uid']) && !empty($_REQUEST['uid'])) {
                $exIds = explode(,, $_REQUEST['uid']);
                $out = array();

                foreach ($exIds as $id) {
                    $e = new Email();
                    $e->retrieve($id);
                    $e->description_html = from_html($e->description_html);
                    $ie->email = $e;
                    $out[] = $ie->displayOneEmail($id, $_REQUEST['mbox']);
                }

                echo $json->encode($out);
            }

            break;


// lots of code
// on line 1082
        case refreshSugarFolders:
            try {
                $GLOBALS['log']->debug(********** EMAIL 2.0 - Asynchronous - at:␣
↪refreshSugarFolders);
                $rootNode = new ExtNode('', '');
                $folderOpenState = $current_user->getPreference('folderOpenState',
↪'Emails');
                $folderOpenState = (empty($folderOpenState)) ?  : $folderOpenState;
                $ret = $email->et->folder->getUserFolders(
                    $rootNode,
                    sugar_unserialize($folderOpenState),
                    $current_user,
                    true
                );
                $out = $json->encode($ret);
                echo $out;
            } catch (SugarFolderEmptyException $e) {
                $GLOBALS['log']->warn($e);
                $out = $json->encode(array(
                    'message' => 'No folder selected warning message here...',
                ));
                echo $out;
            }
            break;
```

10.2.55 Deep Definitions

Dolphin

Deep Definitions, in wp-admin/includes/misc.php:74.

The ConstructHiddenValues function builds the ConstructHiddenSubValues function. Thus, ConstructHiddenValues can only be called once.

```
function ConstructHiddenValues($Values)
{
    /**
     *    Recursive function, processes multidimensional arrays
     *
     * @param string $Name  Full name of array, including all subarrays' names
     *
     * @param array  $Value Array of values, can be multidimensional
     *
     * @return string    Properly consctructed <input type="hidden"...> tags
     */
    function ConstructHiddenSubValues($Name, $Value)
    {
        if (is_array($Value)) {
            $Result = "";
            foreach ($Value as $KeyName => $SubValue) {
                $Result .= ConstructHiddenSubValues("{$Name}[{$KeyName}]", $SubValue);
            }
        } else // Exit recurse
        {
            $Result = "<input type=\"hidden\" name=\"" . htmlspecialchars($Name) . "\
↪" value=\"" . htmlspecialchars($Value) . "\" />\n";
        }

        return $Result;
    }

    /* End of ConstructHiddenSubValues function */

    $Result = '';
    if (is_array($Values)) {
        foreach ($Values as $KeyName => $Value) {
            $Result .= ConstructHiddenSubValues($KeyName, $Value);
        }
    }

    return $Result;
}
```

10.2.56 Repeated print()

Edusoho

Repeated print(), in app/check.php:71.

All echo may be merged into one : do this by turning the ; and . into ‘,’, and removing the superfluous echo. Also, echo_style may be turned into a non-display function, returning the build style, rather than echoing it to the output.

```
echo PHP_EOL;
echo_style('title', 'Note');
echo '  The command console could use a different php.ini file'.PHP_EOL;
echo_style('title', '~~~~');
echo '  than the one used with your web server. To be on the'.PHP_EOL;
echo '      safe side, please check the requirements from your web'.PHP_EOL;
echo '      server using the ';
echo_style('yellow', 'web/config.php');
echo ' script.'.PHP_EOL;
echo PHP_EOL;
```

HuMo-Gen

Repeated print(), in menu.php:71.

Simply calling print once is better than three times. Here too, echo usage would reduce the amount of memory allocation due to concatenation prior display.

```
print '<input type=text name=quicksearch value=.$quicksearch. size=10 '.$pattern.'␣
↪title=.__(Minimum:).$min_chars.__(characters).>';
                    print ' <input type=submit value=.__(Search).>';
            print </form>;
```

10.2.57 Objects Don't Need References

Zencart

Objects Don't Need References, in includes/library/illuminate/support/helpers.php:484.

No need for & operator when $class is only used for a method call.

```
/**
     * @param $class
     * @param $eventID
     * @param array $paramsArray
     */
    public function updateNotifyCheckoutflowFinishedManageSuccessOrderLinkEnd(&$class,
↪ $eventID, $paramsArray = array())
    {
        $class->getView()->getTplVarManager()->se('flag_show_order_link', false);
    }
```

XOOPS

Objects Don't Need References, in htdocs/class/theme_blocks.phps:221.

Here, $template is modified, when its properties are modified. When only the properties are modified, or read, then & is not necessary.

```
public function buildBlock($xobject, &$template)
    {
        // The lame type workaround will change
        // bid is added temporarily as workaround for specific block manipulation
        $block = array(
            'id'      => $xobject->getVar('bid'),
            'module'  => $xobject->getVar('dirname'),
            'title'   => $xobject->getVar('title'),
            // 'name'        => strtolower( preg_replace( '/[^0-9a-zA-Z_]/', '', str_
↪replace( ' ', '_', $xobject->getVar( 'name' ) ) ) ),
            'weight'  => $xobject->getVar('weight'),
            'lastmod' => $xobject->getVar('last_modified'));

        $bcachetime = (int)$xobject->getVar('bcachetime');
        if (empty($bcachetime)) {
            $template->caching = 0;
        } else {
            $template->caching        = 2;
            $template->cache_lifetime = $bcachetime;
        }
        $template->setCompileId($xobject->getVar('dirname', 'n'));
        $tplName = ($tplName = $xobject->getVar('template')) ? db:$tplName :
↪'db:system_block_dummy.tpl';
        $cacheid = $this->generateCacheId('blk_' . $xobject->getVar('bid'));
// more code to the end of the method
```

10.2.58 Lost References

WordPress

Lost References, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

10.2.59 Never Used Properties

WordPress

Never Used Properties, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

10.2.60 No Real Comparison

Magento

No Real Comparison, in app/code/core/Mage/XmlConnect/Block/Catalog/Product/Options/Configurable.php:74.

Compare prices and physical quantities with a difference, so as to avoid rounding errors.

```
if ((float)$option['price'] != 0.00) {
                        $valueNode->addAttribute('price', $option['price']);
                        $valueNode->addAttribute('formated_price', $option['formated_
↪price']);
                    }
```

SPIP

No Real Comparison, in ecrire/maj/v017.php:37.

Here, the current version number is stored as a real number. With a string, though a longer value, it may be compared using the version_compare() function.

```
$version_installee == 1.701
```

10.2.61 Unused Global

Dolphin

Unused Global, in Dolphin-v.7.3.5/modules/boonex/forum/classes/DbForum.php:548.

$gConf is not used in this method, and may be safely avoided.

```
function getUserPostsList ($user, $sort, $limit = 10)
    {
        global $gConf;

        switch ($sort) {
            case 'top':
                $order_by = " t1.`votes` DESC ";
                break;
            case 'rnd':
                $order_by = " RAND() ";
                break;
            default:
                $order_by = " t1.`when` DESC ";
        }

        $sql =  "
        SELECT t1.`forum_id`, t1.`topic_id`, t2.`topic_uri`, t2.`topic_title`, t1.
↪`post_id`, t1.`user`, `post_text`, t1.`when`
            FROM " . TF_FORUM_POST . " AS t1
        INNER JOIN " . TF_FORUM_TOPIC . " AS t2
            ON (t1.`topic_id` = t2.`topic_id`)
        WHERE  t1.`user` = '$user' AND `t2`.`topic_hidden` = '0'
        ORDER BY " . $order_by . "
        LIMIT $limit";

        $a = $this->getAll ($sql);
        $this->_cutPostText($a);
        return $a;
    }
```

10.2.62 Useless Global

Zencart

Useless Global, in admin/includes/modules/newsletters/newsletter.php:25.

$_GET is always a global variable. There is no need to declare it global in any scope.

```
function choose_audience() {
        global $_GET;
```

HuMo-Gen

Useless Global, in relations.php:332.

It is hard to spot that $generY is useless, but this is the only occurrence where $generY is refered to as a global. It is not accessed anywhere else as a global (there are occurrences of $generY being an argument), and it is not even assigned within that function.

```
function calculate_ancestor($pers) {
    global $db_functions, $reltext, $sexe, $sexe2, $spouse, $special_spouseY,
↪$language, $ancestortext, $dutchtext, $selected_language, $spantext, $generY,
↪$foundY_nr, $rel_arrayY;
```

10.2.63 Preprocessable

phpadsnew

Preprocessable, in phpAdsNew-2.0/adview.php:302.

Each call to chr() may be done before. First, chr() may be replace with the hexadecimal sequence “0x3B”; Secondly, 0x3b is a rather long replacement for a simple semi-colon. The whole pragraph could be stored in a separate file, for easier modifications.

```
echo chr(0x47).chr(0x49).chr(0x46).chr(0x38).chr(0x39).chr(0x61).chr(0x01).chr(0x00).
                 chr(0x01).chr(0x00).chr(0x80).chr(0x00).chr(0x00).chr(0x04).
↪chr(0x02).chr(0x04).
                     chr(0x00).chr(0x00).chr(0x00).chr(0x21).chr(0xF9).chr(0x04).
↪chr(0x01).chr(0x00).
                 chr(0x00).chr(0x00).chr(0x00).chr(0x2C).chr(0x00).chr(0x00).
↪chr(0x00).chr(0x00).
                 chr(0x01).chr(0x00).chr(0x01).chr(0x00).chr(0x00).chr(0x02).
↪chr(0x02).chr(0x44).
                 chr(0x01).chr(0x00).chr(0x3B);
```

10.2.64 Useless Unset

Tine20

Useless Unset, in tine20/Felamimail/Controller/Message.php:542.

$_rawContent is unset after being sent to the stream. The variable is a parameter, and will be freed at the end of the call of the method. No need to do it explicitly.

```
protected function _createMimePart($_rawContent, $_partStructure)
    {
        if (Tinebase_Core::isLogLevel(Zend_Log::TRACE)) Tinebase_Core::getLogger()->
↪trace(__METHOD__ . '::' . __LINE__ . ' Content: ' . $_rawContent);

        $stream = fopen(php://temp, 'r+');
        fputs($stream, $_rawContent);
        rewind($stream);

        unset($_rawContent);
        //..... More code, no usage of $_rawContent
    }
```

Typo3

Useless Unset, in typo3/sysext/frontend/Classes/Page/PageRepository.php:708.

$row is unset under certain conditions : here, we can read it in the comments. Eventually, the $row will be returned, and turned into a NULL, by default. This will also create a notice in the logs. Here, the best would be to set a null value, instead of unsetting the variable.

```
public function getRecordOverlay($table, $row, $sys_language_content, $OLmode = '')
    {
//....  a lot more code, with usage of $row, and several unset($row)
//...... Reduced for simplicity
                    } else {
                        // When default language is displayed, we never want to␣
↪return a record carrying
                        // another language!
                        if ($row[$GLOBALS['TCA'][$table]['ctrl']['languageField']] >␣
↪0) {
                            unset($row);
                        }
                    }
                }
            }
        }
        foreach ($GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']['t3lib/class.t3lib_page.php
↪']['getRecordOverlay'] ?? [] as $className) {
            $hookObject = GeneralUtility::makeInstance($className);
            if (!$hookObject instanceof PageRepositoryGetRecordOverlayHookInterface) {
                throw new \UnexpectedValueException($className . ' must implement␣
↪interface ' . PageRepositoryGetRecordOverlayHookInterface::class, 1269881659);
            }
            $hookObject->getRecordOverlay_postProcess($table, $row, $sys_language_
↪content, $OLmode, $this);
        }
        return $row;
    }
```

10.2.65 Buried Assignation

XOOPS

Buried Assignation, in htdocs/image.php:170.

Classic iffectation : the condition also collects the needed value to process the drawing. This is very common in PHP, and the Yoda condition, with its constant on the left, shows that extra steps were taken to strengthen that piece of code.

```
if (0 < ($radius = $radii[2] * $q)) { // left bottom
        imagearc($workingImage, $radius - 1, $workingHeight - $radius, $radius * 2,
↪$radius * 2, 90, 180, $alphaColor);
        imagefilltoborder($workingImage, 0, $workingHeight - 1, $alphaColor,
↪$alphaColor);
    }
```

Mautic

Buried Assignation, in app/bundles/CoreBundle/Controller/ThemeController.php:47.

The setting of the variable $cancelled is fairly hidden here, with its extra operator !. The operator is here for the condition, as $cancelled needs the 'cancellation' state, while the condition needs the contrary. Note also that isset() could be moved out of this condition, and made the result easier to read.

```
$form        = $this->get('form.factory')->create('theme_upload', [], ['action' =>
↪$action]);

        if ($this->request->getMethod() == 'POST') {
            if (isset($form) && !$cancelled = $this->isFormCancelled($form)) {
                if ($this->isFormValid($form)) {
                    $fileData = $form['file']->getData();
```

10.2.66 No array_merge() In Loops

Tine20

No array_merge() In Loops, in tine20/Tinebase/User/Ldap.php:670.

Classic example of array_merge() in loop : here, the attributures should be collected in a local variable, and then merged in one operation, at the end. That includes the attributes provided before the loop, and the array provided after the loop. Note that the order of merge will be the same when merging than when collecting the arrays.

```
$attributes = array_values($this->_rowNameMapping);
        foreach ($this->_ldapPlugins as $plugin) {
            $attributes = array_merge($attributes, $plugin->getSupportedAttributes());
        }

        $attributes = array_merge($attributes, $this->_
↪additionalLdapAttributesToFetch);
```

10.2.67 Useless Parenthesis

Mautic

Useless Parenthesis, in code/app/bundles/EmailBundle/Controller/AjaxController.php:85.

Parenthesis are useless around $progress[1], and around the division too.

```
$dataArray['percent'] = ($progress[1]) ? ceil(($progress[0] / $progress[1]) * 100) :␣
↪100;
```

Woocommerce

Useless Parenthesis, in includes/class-wc-coupon.php:437.

Parenthesis are useless for calculating $discount_percent, as it is a division. Moreover, it is not needed with $discount, (float) applies to the next element, but it does make the expression more readable.

```
if ( wc_prices_include_tax() ) {
    $discount_percent = ( wc_get_price_including_tax( $cart_item['data'] ) * $cart_
↪item_qty ) / WC()->cart->subtotal;
} else {
    $discount_percent = ( wc_get_price_excluding_tax( $cart_item['data'] ) * $cart_
↪item_qty ) / WC()->cart->subtotal_ex_tax;
}
$discount = ( (float) $this->get_amount() * $discount_percent ) / $cart_item_qty;
```

10.2.68 Unresolved Instanceof

WordPress

Unresolved Instanceof, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
private function resolveTag($match)
    {
        $tagReflector = $this->createLinkOrSeeTagFromRegexMatch($match);
        if (!$tagReflector instanceof Tag\SeeTag && !$tagReflector instanceof␣
↪Tag\LinkTag) {
            return $match;
        }
```

10.2.69 Use PHP Object API

WordPress

Use PHP Object API, in wp-includes/functions.php:2558.

Finfo has also a class, with the same name.

```
finfo_open(FILEINFO_MIME_TYPE)
```

PrestaShop

Use PHP Object API, in admin-dev/filemanager/include/utils.php:174.

transliterator_transliterate() has also a class named Transliterator

```
transliterator_transliterate('Accents-Any', $str)
```

SugarCrm

Use PHP Object API, in SugarCE-Full-6.5.26/include/database/MysqliManager.php:222.

Mysqli has also a class, with the same name.

```
mysqli_fetch_field_direct($result, $i)
```

10.2.70 Altering Foreach Without Reference

Contao

Altering Foreach Without Reference, in core-bundle/src/Resources/contao/classes/Theme.php:613.

$tmp[$kk] is &$vv.

```
foreach ($tmp as $kk=>$vv)
                                                             {
                                                                     // Do not use the␣
↪FilesModel here - tables are locked!
                                                                     $objFile = $this->
↪Database->prepare(SELECT uuid FROM tl_files WHERE path=?)
                                                                                       ␣
↪                                 ->limit(1)
                                                                                       ␣
↪                                 ->execute($this->customizeUploadPath($vv));
                                                                     $tmp[$kk] =
↪$objFile->uuid;
                                                             }
```

WordPress

Altering Foreach Without Reference, in wp-admin/includes/misc.php:74.

$ids[$index] is &$rrid.

```
foreach($ids as $index => $rrid)
                {
                    if($rrid == $this->Id)
                    {
                        $ids[$index] = $_id;
```

(continues on next page)

(continued from previous page)

```
                        $write = true;
                        break;
                    }
                }
```

10.2.71 Old Style __autoload()

Piwigo

Old Style __autoload(), in include/phpmailer/PHPMailerAutoload.php:45.

This code handles situations for PHP after 5.1.0 and older. Rare are the applications that are still using those versions in 2019.

```
if (version_compare(PHP_VERSION, '5.1.2', '>=')) {
    //SPL autoloading was introduced in PHP 5.1.2
    if (version_compare(PHP_VERSION, '5.3.0', '>=')) {
        spl_autoload_register('PHPMailerAutoload', true, true);
    } else {
        spl_autoload_register('PHPMailerAutoload');
    }
} else {
    /**
     * Fall back to traditional autoload for old PHP versions
     * @param string $classname The name of the class to load
     */
    function __autoload($classname)
    {
        PHPMailerAutoload($classname);
    }
}
```

10.2.72 Empty Instructions

Zurmo

Empty Instructions, in app/protected/core/widgets/MentionInput.php:84.

There is no need for a semi-colon after a if/then structure.

```
public function run()
        {
            $id = $this->getId();
            $additionalSettingsJs = showAvatars: . var_export($this->showAvatars,␣
↪true) . ,;
            if ($this->classes)
            {
                $additionalSettingsJs .=  $this->classes . ',';
            };
            if ($this->templates)
            {
                $additionalSettingsJs .=  $this->templates;
            };
```

ThinkPHP

Empty Instructions, in ThinkPHP/Library/Vendor/Smarty/sysplugins/smarty_internal_configfileparser.php:83.

There is no need for a semi-colon after a class structure, unless it is an anonymous class.

```
class TPC_yyStackEntry
{
    public $stateno;       /* The state-number */
    public $major;         /* The major token value.  This is the code
                     ** number for the token at this stack level */
    public $minor; /* The user-supplied minor token value.  This
                     ** is the value of the token  */
};
```

10.2.73 Use Pathinfo

SuiteCrm

Use Pathinfo, in include/utils/file_utils.php:441.

Looking for the extension ? Use pathinfo() and PATHINFO_EXTENSION

```
$exp = explode('.', $filename);
```

10.2.74 Should Use Constants

Tine20

Should Use Constants, in tine20/Sales/Controller/Invoice.php:560.

True should be replaced by COUNT_RECURSIVE. The default one is COUNT_NORMAL.

```
count($billables, true)
```

10.2.75 No Parenthesis For Language Construct

Phpdocumentor

No Parenthesis For Language Construct, in src/Application/Renderer/Router/StandardRouter.php:55.

No need for parenthesis with require(). instanceof has a higher precedence than return anyway.

```
$this[] = new Rule(function ($node) { return ($node instanceof NamespaceDescriptor); }
↪, $namespaceGenerator);
```

phpMyAdmin

No Parenthesis For Language Construct, in db_datadict.php:170.

Not only echo() doesn't use any parenthesis, but this syntax gives the illusion that echo() only accepts one argument, while it actually accepts an arbitrary number of argument.

```
echo (($row['Null'] == 'NO') ? __('No') : __('Yes'))
```

10.2.76 No Hardcoded Path

Tine20

No Hardcoded Path, in tine20/Tinebase/DummyController.php:28.

When this script is not run on a Linux system, the file save will fail.

```
file_put_contents('/var/run/tine20/DummyController.txt', 'success ' . $n)
```

Thelia

No Hardcoded Path, in local/modules/Tinymce/Resources/js/tinymce/filemanager/include/php_image_magician.php:2317.

The *iptc.jpg* file is written. It looks like the file may be written next to the php_image_magician.php file, but this is deep in the source code and is unlikely. This means that the working directory has been set to some other place, though we don't read it immediately.

```
private function writeIPTC($dat, $value)
    {

            # LIMIT TO JPG

            $caption_block = $this->iptc_maketag(2, $dat, $value);
            $image_string = iptcembed($caption_block, $this->fileName);
            file_put_contents('iptc.jpg', $image_string);
    }
```

10.2.77 No Hardcoded Port

WordPress

No Hardcoded Port, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

10.2.78 Use Constant As Arguments

Tikiwiki

Use Constant As Arguments, in lib/language/Language.php:112.

E_WARNING is a valid value, but PHP documentation for trigger_error() explains that E_USER constants should be used.

```
trigger_error("Octal or hexadecimal string '" . $match[1] . "' not supported", E_
↪WARNING)
```

---

shopware

Use Constant As Arguments, in engine/Shopware/Plugins/Default/Core/Debug/Components/EventCollector.php:106.

One example where code review reports errors where unit tests don't : array_multisort actually requires sort order first (SORT_ASC or SORT_DESC), then sort flags (such as SORT_NUMERIC). Here, with SORT_DESC = 3 and SORT_NUMERIC = 1, PHP understands it as the coders expects it. The same error is repeated six times in the code.

```
array_multisort($order, SORT_NUMERIC, SORT_DESC, $this->results)
```

10.2.79 Assign Default To Properties

LiveZilla

Assign Default To Properties, in livezilla/_lib/functions.external.inc.php:174.

Flags may default to array() in the class definition. Filled array(), with keys and values, are also possible.

```
class OverlayChat
{
    public $Botmode;
    public $Human;
    public $HumanGeneral;
    public $RepollRequired;
    public $OperatorCount;
    public $Flags;
    public $LastMessageReceived;
    public $LastPostReceived;
    public $IsHumanChatAvailable;
    public $IsChatAvailable;
    public $ChatHTML;
    public $OverlayHTML;
    public $PostHTML;
    public $FullLoad;
    public $LanguageRequired = false;
    public $LastPoster;
    public $EyeCatcher;
    public $GroupBuilder;
    public $CurrentOperatorId;
    public $BotTitle;
    public $OperatorPostCount;
```

(continues on next page)

(continued from previous page)

```
    public $PlaySound;
    public $SpeakingToHTML;
    public $SpeakingToAdded;
    public $Version = 1;

    public static $MaxPosts = 50;
    public static $Response;

    function __construct()
    {
        $this->Flags = array();
        VisitorChat::$Router = new ChatRouter();
    }
```

phpMyAdmin

Assign Default To Properties, in libraries/classes/Console.ph:55.

_isEnabled may default to true. It could also default to a class constant.

```
class Console
{
    /**
     * Whether to display anything
     *
     * @access private
     * @var bool
     */
    private $_isEnabled;

// some code ignored here
    /**
     * Creates a new class instance
     */
    public function __construct()
    {
        $this->_isEnabled = true;
```

10.2.80 Should Chain Exception

Magento

Should Chain Exception, in lib/Mage/Backup/Filesystem/Rollback/Ftp.php:81.

Instead of using the exception message as an argument, chaining the exception would send the whole exception, including the message, and other interesting information like file and line.

```
protected function _initFtpClient()
    {
        try {
            $this->_ftpClient = new Mage_System_Ftp();
            $this->_ftpClient->connect($this->_snapshot->getFtpConnectString());
```

(continues on next page)

(continued from previous page)

```
        } catch (Exception $e) {
            throw new Mage_Backup_Exception_FtpConnectionFailed($e->getMessage());
        }
    }
```

Tine20

Should Chain Exception, in tine20/Setup/Controller.php:81.

Here, the new exception gets an hardcoded message. More details about the reasons are already available in the $e exception, but they are not logged, not chained for later processing.

```
try {
            $dirIterator = new DirectoryIterator($this->_baseDir);
        } catch (Exception $e) {
            Setup_Core::getLogger()->warn(__METHOD__ . '::' . __LINE__ . ' Could not␣
↪open base dir: ' . $this->_baseDir);
            throw new Tinebase_Exception_AccessDenied('Could not open Tine 2.0 root␣
↪directory.');
        }
```

10.2.81 Undefined Interfaces

xataface

Undefined Interfaces, in Dataface/Error.php:112.

Exception seems to be a typo, and leads to an always-true expression.

```
public static function isError($obj){
            if ( !PEAR::isError($obj) and !($obj instanceof Exception_) ) return␣
↪false;
            return ($obj->getCode() >= DATAFACE_E_ERROR);
    }
```

10.2.82 Useless Interfaces

Woocommerce

Useless Interfaces, in includes/interfaces/class-wc-order-item-data-store-interface.php:20.

WC_Order_Item_Data_Store_Interface is used to structure the class WC_Order_Item_Data_Store. It is not used anywhere else.

```
interface WC_Order_Item_Data_Store_Interface {


////////
//includes/data-stores/class-wc-order-item-data-store.php

class WC_Order_Item_Data_Store implements WC_Order_Item_Data_Store_Interface {
```

10.2.83 Should Use Prepared Statement

Dolibarr

Should Use Prepared Statement, in htdocs/product/admin/price_rules.php:76.

This code is well escaped, as the integer type cast will prevent any special chars to be used. Here, a prepared statement would apply a modern approach to securing this query.

```
$db->query("DELETE FROM " . MAIN_DB_PREFIX . "product_pricerules WHERE level = " .␣
↪(int) $i)
```

10.2.84 No Hardcoded Ip

OpenEMR

No Hardcoded Ip, in wp-admin/includes/misc.php:74.

Although they are commented just above, the values provided here are suspicious.

```
// FTP parameters that you must customize.  If you are not sending
 // then set $FTP_SERVER to an empty string.
 //
 $FTP_SERVER = 192.168.0.30;
 $FTP_USER   = openemr;
 $FTP_PASS   = secret;
 $FTP_DIR    = ;
```

NextCloud

No Hardcoded Ip, in config/config.sample.php:1561.

Although they are documented as empty array, 3 values are provided as examples. They do not responds, at the time of writing, but they may.

```
/**
 * List of trusted proxy servers
 *
 * You may set this to an array containing a combination of
 * - IPv4 addresses, e.g. `192.168.2.123`
 * - IPv4 ranges in CIDR notation, e.g. `192.168.2.0/24`
 * - IPv6 addresses, e.g. `fd9e:21a7:a92c:2323::1`
 *
 * _(CIDR notation for IPv6 is currently work in progress and thus not
 * available as of yet)_
 *
 * When an incoming request's `REMOTE_ADDR` matches any of the IP addresses
 * specified here, it is assumed to be a proxy instead of a client. Thus, the
 * client IP will be read from the HTTP header specified in
 * `forwarded_for_headers` instead of from `REMOTE_ADDR`.
 *
 * So if you configure `trusted_proxies`, also consider setting
 * `forwarded_for_headers` which otherwise defaults to `HTTP_X_FORWARDED_FOR`
```

(continues on next page)

(continued from previous page)

```
  * (the `X-Forwarded-For` header).
  *
  * Defaults to an empty array.
  */
'trusted_proxies' => array('203.0.113.45', '198.51.100.128', '192.168.2.0/24'),
```

10.2.85 Echo With Concat

Phpdocumentor

Echo With Concat, in src/phpDocumentor/Bootstrap.php:76.

Simply replace the dot by a comma.

```
echo 'PROFILING ENABLED' . PHP_EOL
```

TeamPass

Echo With Concat, in includes/libraries/Authentication/Yubico/PEAR.php:162.

This is less obvious, but turning print to echo, and the double-quoted string to single quoted string will yield the same optimisation.

```
print "PEAR constructor called, class=$classname\n";
```

10.2.86 Else If Versus Elseif

TeamPass

Else If Versus Elseif, in items.php:819.

This code could be turned into a switch() structure.

```
if ($field[3] === 'text') {
                echo '
                        <input type=text id=edit_field_.$field[0]._.$elem[0].␣
↪class=edit_item_field input_text text ui-widget-content ui-corner-all size=40 data-
↪field-type=.$field[3]. data-field-masked=.$field[4]. data-field-is-mandatory=.
↪$field[5]. data-template-id=.$templateID.>';
            } else if ($field[3] === 'textarea') {
                echo '
                        <textarea id=edit_field_.$field[0]._.$elem[0]. class=edit_
↪item_field input_text text ui-widget-content ui-corner-all colums=40 rows=5 data-
↪field-type=.$field["3"]. data-field-masked=.$field[4]. data-field-is-mandatory=.
↪$field[5]. data-template-id=.$templateID.></textarea>';
            }
```

Phpdocumentor

Else If Versus Elseif, in src/phpDocumentor/Plugin/Core/Transformer/Writer/Xsl.php:112.

The first then block is long and complex. The else block, on the other hand, only contains a single if/then/else. Both conditions are distinct at first sight, so a if / elseif / then structure would be the best.

```
if ($transformation->getQuery() !== '') {
/** Long then block **/
        } else {
            if (substr($transformation->getArtifact(), 0, 1) == '$') {
                // not a file, it must become a variable!
                $variable_name = substr($transformation->getArtifact(), 1);
                $this->xsl_variables[$variable_name] = $proc->transformToXml(
↪$structure);
            } else {
                $relativeFileName = substr($artifact, strlen($transformation->
↪getTransformer()->getTarget()) + 1);
                $proc->setParameter('', 'root', str_repeat('../', substr_count(
↪$relativeFileName, '/')));

                $this->writeToFile($artifact, $proc, $structure);
            }
        }
```

10.2.87 Could Be Static

Dolphin

Could Be Static, in inc/utils.inc.php:673.

Dolphin pro relies on HTMLPurifier to handle cleaning of values : it is used to prevent xss threat. In this method, oHtmlPurifier is first checked, and if needed, created. Since creation is long and costly, it is only created once. Once the object is created, it is stored as a global to be accessible at the next call of the method. In fact, oHtmlPurifier is never used outside this method, so it could be turned into a 'static' variable, and prevent other methods to modify it. This is a typical example of variable that could be static instead of global.

```
function clear_xss($val)
{
    // HTML Purifier plugin
    global $oHtmlPurifier;
    if (!isset($oHtmlPurifier) && !$GLOBALS['logged']['admin']) {

        require_once(BX_DIRECTORY_PATH_PLUGINS . 'htmlpurifier/HTMLPurifier.
↪standalone.php');

/..../

        $oHtmlPurifier = new HTMLPurifier($oConfig);
    }

    if (!$GLOBALS['logged']['admin']) {
        $val = $oHtmlPurifier->purify($val);
    }

    $oZ = new BxDolAlerts('system', 'clear_xss', 0, 0,
```

(continues on next page)

(continued from previous page)

```
        array('oHtmlPurifier' => $oHtmlPurifier, 'return_data' => &$val));
    $oZ->alert();

    return $val;
}
```

Contao

Could Be Static, in system/helper/functions.php:184.

$arrScanCache is a typical cache variables. It is set as global for persistence between calls. If it contains an already stored answer, it is returned immediately. If it is not set yet, it is then filled with a value, and later reused. This global could be turned into static, and avoid pollution of global space.

```
function scan($strFolder, $blnUncached=false)
{
    global $arrScanCache;

    // Add a trailing slash
    if (substr($strFolder, -1, 1) != '/')
    {
            $strFolder .= '/';
    }

    // Load from cache
    if (!$blnUncached && isset($arrScanCache[$strFolder]))
    {
            return $arrScanCache[$strFolder];
    }
    $arrReturn = array();

    // Scan directory
    foreach (scandir($strFolder) as $strFile)
    {
            if ($strFile == '.' || $strFile == '..')
            {
                    continue;
            }

            $arrReturn[] = $strFile;
    }

    // Cache the result
    if (!$blnUncached)
    {
            $arrScanCache[$strFolder] = $arrReturn;
    }

    return $arrReturn;
}
```

10.2.88 Could Use Short Assignation

ChurchCRM

Could Use Short Assignation, in src/ChurchCRM/utils/GeoUtils.php:74.

Sometimes, the variable is on the other side of the operator.

```
$distance = 0.6213712 * $distance;
```

Thelia

Could Use Short Assignation, in local/modules/Tinymce/Resources/js/tinymce/filemanager/include/utils.php:70.

/= is rare, but it definitely could be used here.

```
$size = $size / 1024;
```

10.2.89 Pre-increment

ExpressionEngine

Pre-increment, in system/ee/EllisLab/ExpressionEngine/Controller/Utilities/Communicate.php:650.

Using preincrement in for() loops is safe and straightforward.

```
for ($x = 0; $x < $number_to_send; $x++)
            {
                    $email_address = array_shift($recipient_array);

                    if ( ! $this->deliverEmail($email, $email_address))
                    {
                            $email->delete();

                            $debug_msg = ee()->email->print_debugger(array());

                            show_error(lang('error_sending_email').BR.BR.$debug_msg);
                    }
                    $email->total_sent++;
            }
```

Traq

Pre-increment, in src/Controllers/Tickets.php:84.

$this->currentProject->next_ticket_id value is ignored by the code. It may be turned into a preincrement.

```
TimelineModel::newTicketEvent($this->currentUser, $ticket)->save();

            $this->currentProject->next_ticket_id++;
            $this->currentProject->save();
```

10.2.90 Indices Are Int Or String

Zencart

Indices Are Int Or String, in includes/modules/payment/paypaldp.php:2523.

All those strings ends up as integers.

```
// Build Currency format table
    $curFormat = Array();
    $curFormat[036]=2;
    $curFormat[124]=2;
    $curFormat[203]=2;
    $curFormat[208]=2;
    $curFormat[348]=2;
    $curFormat[392]=0;
    $curFormat[554]=2;
    $curFormat[578]=2;
    $curFormat[702]=2;
    $curFormat[752]=2;
    $curFormat[756]=2;
    $curFormat[826]=2;
    $curFormat[840]=2;
    $curFormat[978]=2;
    $curFormat[985]=2;
```

Mautic

Indices Are Int Or String, in app/bundles/CoreBundle/Entity/CommonRepository.php:315.

$baseCols has 1 and 0 (respectively) for index.

```
foreach ($metadata->getAssociationMappings() as $field => $association) {
                    if (in_array($association['type'], [ClassMetadataInfo::ONE_TO_ONE,
↪ ClassMetadataInfo::MANY_TO_ONE])) {
                        $baseCols[true][$entityClass][]  = $association['joinColumns
↪'][0]['name'];
                        $baseCols[false][$entityClass][] = $field;
                    }
                }
```

10.2.91 Should Typecast

xataface

Should Typecast, in Dataface/Relationship.php:1612.

This is an exact example. A little further, the same applies to intval($max))

```
intval($min);
```

OpenConf

Should Typecast, in author/upload.php:62.

This is another exact example.

```
intval($_POST['pid']);
```

10.2.92 No Direct Usage

Edusoho

No Direct Usage, in edusoho/src/AppBundle/Controller/Admin/FinanceSettingController.php:107.

Glob() returns false, in case of error. It returns an empty array in case everything is fine, but nothing was found. In case of error, array_map() will stop the script.

```
array_map('unlink', glob($dir.'/MP_verify_*.txt'));
```

XOOPS

No Direct Usage, in htdocs/Frameworks/moduleclasses/moduleadmin/moduleadmin.php:585.

Although the file is readable, file() may return false in case of failure. On the other hand, implode doesn't accept boolean values.

```
$file = XOOPS_ROOT_PATH . /modules/{$module_dir}/docs/changelog.txt;
            if ( is_readable( $file ) ) {
                $ret .= implode( '<br>', file( $file ) ) . \n;
            }
```

10.2.93 Avoid Substr() One

ChurchCRM

Avoid Substr() One, in src/Login.php:141.

No need to call substr() to get only one char.

```
if (substr($LocationFromGet, 0, 1) == "/") {
    $LocationFromGet = substr($LocationFromGet, 1);
}
```

LiveZilla

Avoid Substr() One, in livezilla/_lib/objects.global.inc.php:2243.

No need to call substr() to get only one char.

```
$_hex = str_replace("#", "", $_hex);
            if(strlen($_hex) == 3) {
            $r = hexdec(substr($_hex,0,1).substr($_hex,0,1));
            $g = hexdec(substr($_hex,1,1).substr($_hex,1,1));
            $b = hexdec(substr($_hex,2,1).substr($_hex,2,1));
        } else {
            $r = hexdec(substr($_hex,0,2));
            $g = hexdec(substr($_hex,2,2));
            $b = hexdec(substr($_hex,4,2));
        }
        $rgb = array($r, $g, $b);
        return $rgb;
```

10.2.94 Useless Brackets

ChurchCRM

Useless Brackets, in src/Menu.php:72.

Difficut to guess what was before the block here. It doesn't have any usage for control flow.

```
$new_row = false;
        $count_people = 0;

        {
            foreach ($peopleWithBirthDays as $peopleWithBirthDay) {
                if ($new_row == false) {
                    ?>

                    <div class=row>
                <?php
                    $new_row = true;
                } ?>
                <div class=col-sm-3>
```

Piwigo

Useless Brackets, in picture.php:342.

There is no need for block braces with case. In fact, it does give a false sense of break, while the case will still fall over to the next one.

```
case 'rate' :
    {
      include_once(PHPWG_ROOT_PATH.'include/functions_rate.inc.php');
      rate_picture($page['image_id'], $_POST['rate']);
      redirect($url_self);
    }
```

10.2.95 preg_replace With Option e

Edusoho

preg_replace With Option e, in vendor_user/uc_client/lib/uccode.class.php:32.

This call extract text between [code] tags, then process it with $this->codedisp() and nest it again in the original string. preg_replace_callback() is a drop-in replacement for this piece of code.

```
$message = preg_replace("/\s*\[code\](.+?)\[\/code\]\s*/ies", "$this->codedisp('\1')",
↪ $message);
```

10.2.96 eval() Without Try

FuelCMS

eval() Without Try, in fuel/modules/fuel/controllers/Blocks.php:268.

The @ will prevent any error, while the try/catch allows the processing of certain types of error, namely the Fatal ones.

```
@eval($_name_var_eval)
```

ExpressionEngine

eval() Without Try, in system/ee/EllisLab/Addons/member/mod.member_memberlist.php:637.

$cond is build from values extracted from the $fields array. Although it is probably reasonably safe, a try/catch here will collect any unexpected situation cleaningly.

```
elseif (isset($fields[$val['3']]))
                                    {
                                            if (array_key_exists('m_field_id_'.
↪$fields[$val['3']], $row))
                                            {
                                                    $v = $row['m_field_id_'.$fields[
↪$val['3']]];

                                                    $lcond = str_replace($val['3'], "
↪$v", $lcond);
                                                    $cond = $lcond.' '.$rcond;
                                                    $cond = str_replace("\|", "|",
↪$cond);

                                                    eval("$result = ".$cond.";");
```

10.2.97 Relay Function

TeamPass

Relay Function, in includes/libraries/Goodby/CSV/Import/Standard/Interpreter.php:88.

This example puts actually a name on the events : this method 'delegate' and it does it in the smallest amount of possible work, being given all the arguments.

```
/**
     * delegate to observer
     *
     * @param $observer
     * @param $line
     */
    private function delegate($observer, $line)
    {
        call_user_func($observer, $line);
    }
```

SPIP

Relay Function, in ecrire/inc/json.php:73.

var2js() acts as an alternative for json_encode(). Yet, it used to be directly called by the framework's code and difficult to change. With the advent of json_encode, the native function has been used, and even, a compatibility tool was set up. Thus, the relay function.

```
if (!function_exists('json_encode')) {
    function json_encode($v) {
            return var2js($v);
    }
}
```

10.2.98 Silently Cast Integer

MediaWiki

Silently Cast Integer, in includes/debug/logger/monolog/AvroFormatter.php:167.

Too many ff in the masks.

```
private function encodeLong( $id ) {
            $high   = ( $id & 0xffffffff00000000 ) >> 32;
            $low    = $id & 0x00000000ffffffff;
            return pack( 'NN', $high, $low );
    }
```

10.2.99 Timestamp Difference

Zurmo

Timestamp Difference, in app/protected/modules/import/jobs/ImportCleanupJob.php:73.

This is wrong twice a year, in countries that has day-ligth saving time. One of the weeks will be too short, and the other will be too long.

```
/**
         * Get all imports where the modifiedDateTime was more than 1 week ago.  Then
         * delete the imports.
         * (non-PHPdoc)
         * @see BaseJob::run()
         */
        public function run()
        {
            $oneWeekAgoTimeStamp =␣
↪DateTimeUtil::convertTimestampToDbFormatDateTime(time() - 60 * 60 *24 * 7);
```

shopware

Timestamp Difference, in engine/Shopware/Controllers/Backend/Newsletter.php:150.

When daylight saving strike, the email may suddenly be locked for 1 hour minus 30 seconds ago. The lock will be set for the rest of the hour, until the server catch up.

```
// Check lock time. Add a buffer of 30 seconds to the lock time (default request time)
            if (!empty($mailing['locked']) && strtotime($mailing['locked']) > time() -
↪ 30) {
                echo "Current mail: '" . $subjectCurrentMailing . "'\n";
                echo "Wait " . (strtotime($mailing['locked']) + 30 - time()) . "␣
↪seconds ...\n";
                return;
            }
```

10.2.100 Unused Arguments

ThinkPHP

Unused Arguments, in ThinkPHP/Library/Behavior/AgentCheckBehavior.class.php:18.

$params are requested, but never used. The method is not overloading another one, as the class doesn't extends anything. $params is unused.

```
class AgentCheckBehavior
{
    public function run(&$params)
    {
        //
        $limitProxyVisit = C('LIMIT_PROXY_VISIT', null, true);
        if ($limitProxyVisit && ($_SERVER['HTTP_X_FORWARDED_FOR'] || $_SERVER['HTTP_
↪VIA'] || $_SERVER['HTTP_PROXY_CONNECTION'] || $_SERVER['HTTP_USER_AGENT_VIA'])) {
            //
            exit('Access Denied');
        }
    }
}
```

phpMyAdmin

Unused Arguments, in libraries/classes/Display/Results.php:1985.

Although $column_index is documented, it is not found in the rest of the (long) body of the function. It might have been refactored into $sorted_column_index.

```
/**
     * Prepare parameters and html for sorted table header fields
     *
     * @param array    $sort_expression             sort expression
     * @param array    $sort_expression_nodirection sort expression without direction
     * @param string   $sort_tbl                    The name of the table to which
     *                                             the current column belongs to
     * @param string   $name_to_use_in_sort         The current column under
     *                                             consideration
     * @param array    $sort_direction              sort direction
     * @param stdClass $fields_meta                 set of field properties
     * @param integer  $column_index                The index number to current column
     *
     * @return  array   3 element array - $single_sort_order, $sort_order, $order_img
     *
     * @access  private
     *
     * @see     _getOrderLinkAndSortedHeaderHtml()
     */
    private function _getSingleAndMultiSortUrls(
        array $sort_expression,
        array $sort_expression_nodirection,
        $sort_tbl,
        $name_to_use_in_sort,
        array $sort_direction,
        $fields_meta,
        $column_index
    ) {
    /**/
        // find the sorted column index in row result
        // (this might be a multi-table query)
        $sorted_column_index = false;
    /**/
    }
```

10.2.101 Switch To Switch

Thelia

Switch To Switch, in core/lib/Thelia/Controller/Admin/TranslationsController.php:100.

The two first comparison may be turned into a case, and the last one could be default, or default with a check on empty().

```
if($modulePart == 'core') { /**/ } elseif($modulePart == 'admin-includes') { /**/ }␣
↪elseif(!empty($modulePart)) { /**/ }
```

XOOPS

Switch To Switch, in htdocs/search.php:74.

Here, converting this structure to switch requires to drop the === usage. Also, no default usage here.

```
if($action === 'results') { /**/ } elseif($action === 'showall') { /**/ } elseif(
↪$action === 'showallbyuser') { /**/ }
```

10.2.102 Wrong Parameter Type

Zencart

Wrong Parameter Type, in admin/includes/header.php:180.

setlocale() may be called with null or ‘’ (empty string), and will set values from the environment. When called with “0” (the string), it only reports the current setting. Using an integer is probably undocumented behavior, and falls back to the zero string.

```
$loc = setlocale(LC_TIME, 0);
        if ($loc !== FALSE) echo ' - ' . $loc; //what is the locale in use?
```

10.2.103 Redefined Default

Piwigo

Redefined Default, in admin/include/updates.class.php:34.

default_themes is defined as an empty array, then filled with new values. Same for default_plugins. Both may be defined as declaration time, and not during the constructor.

```
class updates
{
  var $types = array();
  var $plugins;
  var $themes;
  var $languages;
  var $missing = array();
  var $default_plugins = array();
  var $default_themes = array();
  var $default_languages = array();
  var $merged_extensions = array();
  var $merged_extension_url = 'http://piwigo.org/download/merged_extensions.txt';

  function __construct($page='updates')
  {
    $this->types = array('plugins', 'themes', 'languages');

    if (in_array($page, $this->types))
    {
      $this->types = array($page);
    }
    $this->default_themes = array('clear', 'dark', 'Sylvia', 'elegant', 'smartpocket
↪');
    $this->default_plugins = array('AdminTools', 'TakeATour', 'language_switch',
↪'LocalFilesEditor');
```

(continues on next page)

(continued from previous page)

10.2.104 Wrong fopen() Mode

Tikiwiki

Wrong fopen() Mode, in lib/tikilib.php:6777.

This fopen() mode doesn’t exists. Use ‘w’ instead.

```
fopen('php://temp', 'rw');
```

HuMo-Gen

Wrong fopen() Mode, in include/phprtflite/lib/PHPRtfLite/StreamOutput.php:77.

This fopen() mode doesn’t exists. Use ‘w’ instead.

```
fopen($this->_filename, 'wr', false)
```

10.2.105 Use random_int()

Thelia

Use random_int(), in core/lib/Thelia/Tools/TokenProvider.php:151.

The whole function may be replaced by random_int(), as it generates random tokens. This needs an extra layer of hashing, to get a long and string results.

```
/**
     * @return string
     */
    protected static function getComplexRandom()
    {
        $firstValue = (float) (mt_rand(1, 0xFFFF) * rand(1, 0x10001));
        $secondValues = (float) (rand(1, 0xFFFF) * mt_rand(1, 0x10001));

        return microtime() . ceil($firstValue / $secondValues) . uniqid();
    }
```

FuelCMS

Use random_int(), in fuel/modules/fuel/libraries/Fuel.php:235.

Security tokens should be build with a CSPRNG source. uniqid() is based on time, and though it changes anytime (sic), it is easy to guess. Those days, it looks like ‘5b1262e74dbb9’;

```
$this->installer->change_config('config', '$config[\'encryption_key\'] = \'\';', '
↪$config[\'encryption_key\'] = \''.md5(uniqid()).'\';');
```

10.2.106 Already Parents Interface

WordPress

Already Parents Interface, in src/Phinx/Db/Adapter/AbstractAdapter.php:41.

SqlServerAdapter extends PdoAdapter, PdoAdapter extends AbstractAdapter. The first and the last both implements AdapterInterface. Only one is needed.

```
/**
 * Base Abstract Database Adapter.
 */
abstract class AbstractAdapter implements AdapterInterface
{

/// In the src/src/Phinx/Db/Adapter/SqlServerAdapter.php, line 45
/**
 * Phinx SqlServer Adapter.
 *
 */
class SqlServerAdapter extends PdoAdapter implements AdapterInterface
{
```

Thelia

Already Parents Interface, in core/lib/Thelia/Core/Template/Loop/BaseSpecificModule.php:35.

PropelSearchLoopInterface is implemented by both BaseSpecificModule and Payment

```
abstract class BaseSpecificModule extends BaseI18nLoop implements␣
↪PropelSearchLoopInterface

/* in file  core/lib/Thelia/Core/Template/Loop/Payment.php, line 28 */

class Payment extends BaseSpecificModule implements PropelSearchLoopInterface
```

10.2.107 Ternary In Concat

TeamPass

Ternary In Concat, in includes/libraries/protect/AntiXSS/UTF8.php:5409.

The concatenations in the initial comparison are disguised casting. When $str2 is empty too, the ternary operator yields a 0, leading to a systematic failure.

```
$str1 . '' === $str2 . '' ? 0 : strnatcmp(self::strtonatfold($str1),␣
↪self::strtonatfold($str2))
```

10.2.108 No Hardcoded Hash

shopware

No Hardcoded Hash, in engine/Shopware/Models/Document/Data/OrderData.php:254.

This is actually a hashed hardcoded password. As the file explains, this is a demo order, for populating the database when in demo mode, so this is fine. We also learn that the password are securily sorted here. It may also be advised to avoid hardcoding this password, as any demo shop has the same user credential : it is the first to be tried when a demo installation is found.

```
'_userID' => '3',
    '_user' => new ArrayObject([
            'id' => '3',
            'password' => '$2y$10$GAGAC6.1kMRvN4RRcLrYleDx.EfWhHcW./cmoOQg11sjFUY73SO.
↪C',
            'encoder' => 'bcrypt',
            'email' => 'demo@shopware.com',
            'customernumber' => '20005',
```

SugarCrm

No Hardcoded Hash, in SugarCE-Full-6.5.26/include/Smarty/Smarty.class.php:460.

The MD5('Smarty') is hardcoded in the properties. This property is not used in the class, but in parts of the code, when a unique delimiter is needed.

```
/**
     * md5 checksum of the string 'Smarty'
     *
     * @var string
     */
    var $_smarty_md5           = 'f8d698aea36fcbead2b9d5359ffca76f';
```

10.2.109 Identical Conditions

WordPress

Identical Conditions, in wp-admin/theme-editor.php:247.

The condition checks first if $has_templates or $theme->parent(), and one of the two is sufficient to be valid. Then, it checks again that $theme->parent() is activated with &&. This condition may be reduced by calling $theme->parent(), as $has_template is unused here.

```
<?php if ( ( $has_templates || $theme->parent() ) && $theme->parent() ) : ?>
```

Dolibarr

Identical Conditions, in htdocs/core/lib/files.lib.php:2052.

Better check twice that $modulepart is really 'apercusupplier_invoice'.

```
$modulepart == 'apercusupplier_invoice' || $modulepart == 'apercusupplier_invoice'
```

Mautic

Identical Conditions, in app/bundles/CoreBundle/Views/Standard/list.html.php:47.

When the line is long, it tends to be more and more difficult to review the values. Here, one of the two first is too many.

```
!empty($permissions[$permissionBase . ':deleteown']) || !empty($permissions[
↪$permissionBase . ':deleteown']) || !empty($permissions[$permissionBase . ':delete
↪'])
```

10.2.110 No Choice

NextCloud

No Choice, in build/integration/features/bootstrap/FilesDropContext.php:71.

Token is checked, but processed in the same way each time. This actual check is done twice, in the same class, in the method droppingFileWith().

```
public function creatingFolderInDrop($folder) {
            $client = new Client();
            $options = [];
            if (count($this->lastShareData->data->element) > 0){
                    $token = $this->lastShareData->data[0]->token;
            } else {
                    $token = $this->lastShareData->data[0]->token;
            }
            $base = substr($this->baseUrl, 0, -4);
            $fullUrl = $base . '/public.php/webdav/' . $folder;

            $options['auth'] = [$token, ''];
```

Zencart

No Choice, in admin/includes/functions/html_output.php:179.

At least, it always choose the most secure way : use SSL.

```
if ($usessl) {
        $form .= zen_href_link($action, $parameters, 'NONSSL');
      } else {
        $form .= zen_href_link($action, $parameters, 'NONSSL');
      }
```

10.2.111 Common Alternatives

Dolibarr

Common Alternatives, in htdocs/admin/facture.php:531.

The opening an closing tag couldd be moved outside the if condition : they are compulsory in both cases.

```
// Active
                                if (in_array($name, $def))
                                {
                                    print '<td class="center">'."\n";
                                    print '<a href="'.$_SERVER["PHP_SELF"].'?
↪action=del&value='.$name.'">';
                                    print img_picto($langs->trans("Enabled"), 'switch_
↪on');
                                    print '</a>';
                                    print '</td>';
                                }
                                else
                                {
                                    print '<td class=center\>'."\n";
                                    print '<a href="'.$_SERVER["PHP_SELF"].'?
↪action=set&value='.$name.'&scan_dir='.$module->scandir.'&label='.urlencode($module->
↪name).'">'.img_picto($langs->trans("SetAsDefault"), 'switch_off').'</a>';
                                    print "</td>";
                                }
```

NextCloud

Common Alternatives, in apps/encryption/lib/KeyManager.php:436.

$shareKey = $this->getShareKey($path, $uid);* is common to all three alternatives. In fact, *$uid = $this->getPublicShareKeyId();* is not common, and that shoul de reviewed, as *$uid will be undefined.

```
if ($this->util->isMasterKeyEnabled()) {
                    $uid = $this->getMasterKeyId();
                    $shareKey = $this->getShareKey($path, $uid);
                    if ($publicAccess) {
                            $privateKey = $this->getSystemPrivateKey($uid);
                            $privateKey = $this->crypt->decryptPrivateKey($privateKey,
↪ $this->getMasterKeyPassword(), $uid);
                    } else {
                            // when logged in, the master key is already decrypted in␣
↪the session
                            $privateKey = $this->session->getPrivateKey();
                    }
            } else if ($publicAccess) {
                    // use public share key for public links
                    $uid = $this->getPublicShareKeyId();
                    $shareKey = $this->getShareKey($path, $uid);
                    $privateKey = $this->keyStorage->getSystemUserKey($this->
↪publicShareKeyId . '.privateKey', Encryption::ID);
                    $privateKey = $this->crypt->decryptPrivateKey($privateKey);
            } else {
```

(continues on next page)

(continued from previous page)

```
                    $shareKey = $this->getShareKey($path, $uid);
                    $privateKey = $this->session->getPrivateKey();
            }
```

10.2.112 Logical Mistakes

Dolibarr

Logical Mistakes, in htdocs/core/lib/admin.lib.php:1165.

This expression is always true. When *$nbtabsql* is *$nbtablib*, the left part is true; When *$nbtabsql* is *$nbtabsqlsort*, the right part is true; When any other value is provided, both operands are true.

```
$nbtablib != $nbtabsql || $nbtabsql != $nbtabsqlsort
```

Cleverstyle

Logical Mistakes, in modules/HybridAuth/Hybrid/Providers/DigitalOcean.php:123.

This expression is always false. When *$data->account->email_verified* is *true*, the right part is false; When *$data->account->email_verified* is *$data->account->email*, the right part is false; The only viable solution is to have ‘ $data->account->email‘true : this is may be the intend it, though it is not easy to understand.

```
TRUE == $data->account->email_verified and $data->account->email == $data->account->
↪email_verified
```

10.2.113 Same Conditions In Condition

TeamPass

Same Conditions In Condition, in sources/identify.php:1096.

$result == 1 is use once in the main if/then, then again the second if/then/elseif structure. Both are incompatible, since, in the else, *$result* will be different from 1.

```
if ($result == 1) {
                $return = "";
                $logError = "";
                $proceedIdentification = true;
                $userPasswordVerified = false;
                unset($_SESSION['hedgeId']);
                unset($_SESSION['flickercode']);
            } else {
                if ($result < -10) {
                    $logError = "ERROR: ".$result;
                } elseif ($result == -4) {
                    $logError = "Wrong response code, no more tries left.";
                } elseif ($result == -3) {
                    $logError = "Wrong response code, try to reenter.";
                } elseif ($result == -2) {
```

(continues on next page)

(continued from previous page)

```
                    $logError = "Timeout. The response code is not valid anymore.";
                } elseif ($result == -1) {
                    $logError = "Security Error. Did you try to verify the response␣
↪from a different computer?";
                } elseif ($result == 1) {
                    $logError = "Authentication successful, response code correct.
                          <br /><br />Authentification Method for SecureBrowser␣
↪updated!";
                    // Add necessary code here for accessing your Business Application
                }
                $return = "agses_error";
                echo '[{"value" : "'.$return.'", "user_admin":"',
                isset($_SESSION['user_admin']) ? $_SESSION['user_admin'] : "",
                '", "initial_url" : "'.@$_SESSION['initial_url'].'",
                "error" : "'.$logError.'"}]';

                exit();
            }
```

Typo3

Same Conditions In Condition, in typo3/sysext/recordlist/Classes/RecordList/DatabaseRecordList.php:1696.

$table == *'pages* is caught initially, and if it fails, it is tested again in the final else. This won't happen.

```
} elseif ($table === 'pages') {
                                $parameters = ['id' => $this->id, 'pagesOnly' => 1,
↪'returnUrl' => GeneralUtility::getIndpEnv('REQUEST_URI')];
                                $href = (string)$uriBuilder->buildUriFromRoute('db_new
↪', $parameters);
                                $icon = '<a class="btn btn-default" href="' .␣
↪htmlspecialchars($href) . '" title="' . htmlspecialchars($lang->getLL('new')) . '">'
                                    . $spriteIcon->render() . '</a>';
                            } else {
                                $params = '&edit[' . $table . '][' . $this->id .
↪']=new';
                                if ($table === 'pages') {
                                    $params .= '&overrideVals[pages][doktype]=' .␣
↪(int)$this->pageRow['doktype'];
                                }
                                $icon = '<a class="btn btn-default" href="#" onclick="
↪' . htmlspecialchars(BackendUtility::editOnClick($params, '', -1))
                                    . '" title="' . htmlspecialchars($lang->getLL('new
↪')) . '">' . $spriteIcon->render() . '</a>';
                            }
```

10.2.114 Return True False

Mautic

Return True False, in app/bundles/LeadBundle/Model/ListModel.php:125.

$isNew could be a typecast.

```
$isNew = ($entity->getId()) ? false : true;
```

FuelCMS

Return True False, in fuel/modules/fuel/helpers/validator_helper.php:254.

If/then is a lot of code to produce a boolean.

```
function length_min($str, $limit = 1)
    {
            if (strlen(strval($str)) < $limit)
            {
                    return FALSE;
            }
            else
            {
                    return TRUE;
            }
    }
```

10.2.115 Useless Switch

Phpdocumentor

Useless Switch, in fuel/modules/fuel/libraries/Inspection.php:349.

This method parses comments. In fact, comments are represented by other tokens, which may be added or removed at time while coding.

```
public function parse_comments($code)
    {
            $comments = array();
            $tokens = token_get_all($code);

            foreach($tokens as $token)
            {
                    switch($token[0])
                    {
                            case T_DOC_COMMENT:
                                    $comments[] = $token[1];
                                    break;
                }
            }
            return $comments;

    }
```

Dolphin

Useless Switch, in Dolphin-v.7.3.5/inc/classes/BxDolModuleDb.php:34.

$aParams is an argument : this code looks like the switch is reserved for future use.

```
function getModulesBy($aParams = array())
    {
            $sMethod = 'getAll';
        $sPostfix = $sWhereClause = "";

        $sOrderClause = "ORDER BY `title`";
        switch($aParams['type']) {
            case 'path':
                    $sMethod = 'getRow';
                $sPostfix .= '_path';
                $sWhereClause .= "AND `path`='" . $aParams['value'] . "'";
                break;
        }
```

10.2.116 Could Use __DIR__

Woocommerce

Could Use __DIR__, in includes/class-wc-api.php:162.

All the 120 occurrences use *dirname(__FILE__)*, and could be upgraded to __DIR__ if backward compatibility to PHP 5.2 is not critical.

```
private function rest_api_includes() {
            // Exception handler.
            include_once dirname( __FILE__ ) . '/api/class-wc-rest-exception.php';

            // Authentication.
            include_once dirname( __FILE__ ) . '/api/class-wc-rest-authentication.php
↪';
```

Piwigo

Could Use __DIR__, in include/random_compat/random.php:50.

dirname(__FILE__) is cached into $RandomCompatDIR, then reused three times. Using __DIR__ would save that detour.

```
$RandomCompatDIR = dirname(__FILE__);

    require_once $RandomCompatDIR.'/byte_safe_strings.php';
    require_once $RandomCompatDIR.'/cast_to_int.php';
    require_once $RandomCompatDIR.'/error_polyfill.php';
```

10.2.117 Should Use Coalesce

ChurchCRM

Should Use Coalesce, in src/ChurchCRM/Service/FinancialService.php:597.

ChurchCRM features 5 old style ternary operators, which are all in this SQL query. ChurchCRM requires PHP 7.0, so a simple code review could remove them all.

```
$sSQL = "INSERT INTO pledge_plg
                    (plg_famID,
                    plg_FYID,
                    plg_date,
                    plg_amount,
                    plg_schedule,
                    plg_method,
                    plg_comment,
                    plg_DateLastEdited,
                    plg_EditedBy,
                    plg_PledgeOrPayment,
                    plg_fundID,
                    plg_depID,
                    plg_CheckNo,
                    plg_scanString,
                    plg_aut_ID,
                    plg_NonDeductible,
                    plg_GroupKey)
                    VALUES ('".
          $payment->FamilyID."','".
          $payment->FYID."','".
          $payment->Date."','".
          $Fund->Amount."','".
          (isset($payment->schedule) ? $payment->schedule : 'NULL')."','".
          $payment->iMethod."','".
          $Fund->Comment."','".
          date('YmdHis')."',".
          $_SESSION['user']->getId().",'".
          $payment->type."',".
          $Fund->FundID.','.
          $payment->DepositID.','.
          (isset($payment->iCheckNo) ? $payment->iCheckNo : 'NULL').",'".
          (isset($payment->tScanString) ? $payment->tScanString : 'NULL')."','".
          (isset($payment->iAutID) ? $payment->iAutID : 'NULL')."','".
          (isset($Fund->NonDeductible) ? $Fund->NonDeductible : 'NULL')."','".
          $sGroupKey."')";
```

Cleverstyle

Should Use Coalesce, in modules/Feedback/index.php:37.

Cleverstyle nests ternary operators when selecting default values. Here, moving some of them to ?? will reduce the code complexity and make it more readable. Cleverstyle requires PHP 7.0 or more recent.

```
$Page->content(
    h::{'cs-form form'}(
            h::{'section.cs-feedback-form article'}(
                    h::{'header h2.cs-text-center'}($L->Feedback).
                    h::{'table.cs-table[center] tr| td'}(
                            [
                                    h::{'cs-input-text input[name=name][required]'}(
                                            [
                                                    'placeholder' => $L->feedback_
↪name,
                                                    'value'       => $User->user() ?
↪$User->username() : (isset($_POST['name']) ? $_POST['name'] : '')
                                            ]
                                    ),
                                    h::{'cs-input-text␣
↪input[type=email][name=email][required]'}(
                                            [
                                                    'placeholder' => $L->feedback_
↪email,
                                                    'value'       => $User->user() ?
↪$User->email : (isset($_POST['email']) ? $_POST['email'] : '')
                                            ]
                                    ),
                                    h::{'cs-textarea[autosize]␣
↪textarea[name=text][required]'}(
                                            [
                                                    'placeholder' => $L->feedback_
↪text,
                                                    'value'       => isset($_POST[
↪'text']) ? $_POST['text'] : ''
                                            ]
                                    ),
                                    h::{'cs-button button[type=submit]'}($L->feedback_
↪send)
                            ]
                    )
            )
    )
);
```

10.2.118 If With Same Conditions

phpMyAdmin

If With Same Conditions, in libraries/classes/Response.php:345.

The first test on $this->_isSuccess settles the situation with _JSON. Then, a second check is made. Both could be merged, also the second one is fairly long (not shown).

```
if ($this->_isSuccess) {
            $this->_JSON['success'] = true;
        } else {
            $this->_JSON['success'] = false;
            $this->_JSON['error']   = $this->_JSON['message'];
            unset($this->_JSON['message']);
        }
```

(continues on next page)

(continued from previous page)

```
        if ($this->_isSuccess) {
```

Phpdocumentor

If With Same Conditions, in src/phpDocumentor/Transformer/Command/Project/TransformCommand.php:239.

$templates is extracted from $input. If it is empty, a second source is polled. Finally, if nothing has worked, a default value is used (‘clean’). In this case, each attempt is an alternative solution to the previous failing call. The second test could be reported on $templatesFromConfig, and not $templates.

```
$templates = $input->getOption('template');
        if (!$templates) {
            /** @var Template[] $templatesFromConfig */
            $templatesFromConfig = $configurationHelper->getConfigValueFromPath(
↪'transformations/templates');
            foreach ($templatesFromConfig as $template) {
                $templates[] = $template->getName();
            }
        }

        if (!$templates) {
            $templates = array('clean');
        }
```

10.2.119 Throw Functioncall

SugarCrm

Throw Functioncall, in include/externalAPI/cmis_repository_wrapper.php:918.

SugarCRM uses exceptions to fill work in progress. Here, we recognize a forgotten ‘new’ that makes throw call a function named ‘Exception’. This fails with a Fatal Error, and doesn’t issue the right messsage. The same error had propgated in the code by copy and paste : it is available 17 times in that same file.

```
function getContentChanges()
    {
        throw Exception("Not Implemented");
    }
```

Zurmo

Throw Functioncall, in app/protected/modules/gamification/rules/collections/GameCollectionRules.php:66.

Other part of the code actually instantiate the exception before throwing it.

```
abstract class GameCollectionRules
    {
        /**
         * @return string
         * @throws NotImplementedException - Implement in children classes
         */
        public static function getType()
        {
            throw NotImplementedException();
        }
```

10.2.120 Use Instanceof

TeamPass

Use Instanceof, in includes/libraries/Database/Meekrodb/db.class.php:506.

In this code, `is_object()` and `instanceof` have the same basic : they both check that $ts is an object. In fact, `instanceof` is more precise, and give more information about the variable.

```
protected function parseTS($ts) {
    if (is_string($ts)) return date('Y-m-d H:i:s', strtotime($ts));
    else if (is_object($ts) && ($ts instanceof DateTime)) return $ts->format('Y-m-d␣
↪H:i:s');
  }
```

Zencart

Use Instanceof, in includes/modules/payment/firstdata_hco.php:104.

In this code, `is_object()` is used to check the status of the order. Possibly, $order is false or null in case of incompatible status. Yet, when $object is an object, and in particular being a global that may be assigned anywhere else in the code, it seems that the method 'update_status' is magically always available. Here, using instance of to make sure that $order is an 'paypal' class, or a 'storepickup' or any of the payment class.

```
function __construct() {
    global $order;

    // more lines, no mention of $order
    if (is_object($order)) $this->update_status();

    // more code
}
```

10.2.121 Always Positive Comparison

Magento

Always Positive Comparison, in app/code/core/Mage/Dataflow/Model/Profile.php:85.

strlen(($actiosXML) will never be negative, and hence, is always false. This exception is never thrown.

```
if (strlen($actionsXML) < 0 &&
        @simplexml_load_string('<data>' . $actionsXML . '</data>', null, LIBXML_
↪NOERROR) === false) {
            Mage::throwException(Mage::helper('dataflow')->__("Actions XML is not␣
↪valid."));
        }
```

10.2.122 Empty Blocks

Cleverstyle

Empty Blocks, in modules/Blogs/api/Controller.php:44.

Else is empty, but commented.

```
public static function posts_get ($Request) {
            $id = $Request->route_ids(0);
            if ($id) {
                    $post = Posts::instance()->get($id);
                    if (!$post) {
                            throw new ExitException(404);
                    }
                    return $post;
            } else {
                    // TODO: implement latest posts
            }
    }
```

PhpIPAM

Empty Blocks, in wp-admin/includes/misc.php:74.

The `then` block is empty and commented : yet, it may have been clearer to make the condition != and omitted the whole empty block.

```
/* checks */
if($_POST['action'] == delete) {
    # no cecks
}
else {
    # remove spaces
    $_POST['name'] = trim($_POST['name']);

    # length > 4 and < 12
    if( (mb_strlen($_POST['name']) < 2) || (mb_strlen($_POST['name']) > 24) )        {
↪$errors[] = _('Name must be between 4 and 24 characters'); }
```

10.2.123 Dependant Trait

Zencart

Dependant Trait, in app/library/zencart/CheckoutFlow/src/AccountFormValidator.php:14.

Note that addressEntries is used, and is also expected to be an array or an object with ArrayAccess. $addressEntries is only defined in a class called ‘Guest’ which is also the only one using that trait. Any other class using the AccountFormValidator trait must define addressEntries.

```
trait AccountFormValidator
{

    abstract protected function getAddressFieldValue($fieldName);

    /**
     * @return bool|int
     */
    protected function errorProcessing()
    {
        $error = false;
        foreach ($this->addressEntries as $fieldName => $fieldDetails) {
            $this->addressEntries[$fieldName]['value'] = $this->getAddressFieldValue(
↪$fieldName);
            $fieldError = $this->processFieldValidator($fieldName, $fieldDetails);
            $this->addressEntries[$fieldName]['error'] = $fieldError;
            $error = $error | $fieldError;
        }
        return $error;
    }
```

10.2.124 Hidden Use Expression

Tikiwiki

Hidden Use Expression, in lib/core/Tiki/Command/DailyReportSendCommand.php:17.

Sneaky error_reporting, hidden among the use calls.

```
namespace Tiki\Command;

use Symfony\Component\Console\Command\Command;
use Symfony\Component\Console\Input\InputArgument;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Input\InputOption;
use Symfony\Component\Console\Output\OutputInterface;
error_reporting(E_ALL);
use TikiLib;
use Reports_Factory;
```

OpenEMR

Hidden Use Expression, in interface/patient_file/summary/browse.php:23.

Use expression is only reached when the csrf token is checked. This probably save some CPU when no csrf is available, but it breaks the readability of the file.

```
<?php
/**
 * Patient selector for insurance gui
 *
 * @package   OpenEMR
 * @link      http://www.open-emr.org
 * @author    Brady Miller <brady.g.miller@gmail.com>
 * @copyright Copyright (c) 2018 Brady Miller <brady.g.miller@gmail.com>
 * @license   https://github.com/openemr/openemr/blob/master/LICENSE GNU General␣
↪Public License 3
 */


require_once(../../globals.php);
require_once($srcdir/patient.inc);
require_once($srcdir/options.inc.php);

if (!empty($_POST)) {
    if (!verifyCsrfToken($_POST[csrf_token_form])) {
        csrfNotVerified();
    }
}

use OpenEMR\Core\Header;
```

10.2.125 Multiple Alias Definitions

ChurchCRM

Multiple Alias Definitions, in Various files:–.

It is actually surprising to find FamilyQuery defined as ChurchCRMBaseFamilyQuery only once, while all other reference are for ChurchCRMFamilyQuery. That lone use is actually useful in the code, so it is not a forgotten refactorisation.

```
use ChurchCRM\Base\FamilyQuery      // in /src/MapUsingGoogle.php:7

use ChurchCRM\FamilyQuery   // in /src/ChurchCRM/Dashboard/EventsDashboardItem.php:8
                            // and 29 other files
```

Phinx

Multiple Alias Definitions, in Various files:–.

One 'Command' is refering to a local Command class, while the other is refering to an imported class. They are all in a similar name space ConsoleCommand.

```
use Phinx\Console\Command                        //in file /src/Phinx/Console/
↪PhinxApplication.php:34
use Symfony\Component\Console\Command\Command        //in file /src/Phinx/Console/
↪Command/Init.php:31
use Symfony\Component\Console\Command\Command        //in file /src/Phinx/Console/
↪Command/AbstractCommand.php:32
```

10.2.126 Nested Ifthen

LiveZilla

Nested Ifthen, in livezilla/_lib/objects.global.inc.php:847.

The first condition is fairly complex, and could also return early. Then, the second nested if could be merged into one : this would reduce the number of nesting, but make the condition higher.

```
if(isset(Server::$Configuration->File["gl_url_detect"]) && !Server::$Configuration->
↪File["gl_url_detect"] && isset(Server::$Configuration->File["gl_url"]) && !
↪empty(Server::$Configuration->File["gl_url"]))
        {
            $url = Server::$Configuration->File["gl_url"];
        }
        else if(isset($_SERVER["HTTP_HOST"]) && !empty($_SERVER["HTTP_HOST"]))
        {
            $host = $_SERVER["HTTP_HOST"];
            $path = $_SERVER["PHP_SELF"];

            if(!empty($path) && !Str::EndsWith(strtolower($path),strtolower($_file)) &
↪& strpos(strtolower($path),strtolower($_file)) !== false)
            {
                if(empty(Server::$Configuration->File["gl_kbmr"]))
                {
                    Logging::DebugLog(serialize($_SERVER));
                    exit("err 888383; can't read $_SERVER[\"HTTP_HOST\"] and $_
↪SERVER[\"PHP_SELF\"]");
                }
            }

            define("LIVEZILLA_DOMAIN",Communication::GetScheme() . $host);
            $url = LIVEZILLA_DOMAIN . str_replace($_file,"",htmlentities($path,ENT_
↪QUOTES,"UTF-8"));
        }
```

MediaWiki

Nested Ifthen, in includes/Linker.php:1493.

There are 5 level of nesting here, from the beginning of the method, down to the last condition. All work on local variables, as it is a static method. May be breaking this into smaller functions would help readability.

```
public static function normalizeSubpageLink( $contextTitle, $target, &$text ) {
            $ret = $target; # default return value is no change

            # Some namespaces don't allow subpages,
            # so only perform processing if subpages are allowed
            if (
                    $contextTitle && MediaWikiServices::getInstance()->
↪getNamespaceInfo()->
                    hasSubpages( $contextTitle->getNamespace() )
            ) {
                    $hash = strpos( $target, '#' );
                    if ( $hash !== false ) {
```

(continues on next page)

(continued from previous page)

```
                           $suffix = substr( $target, $hash );
                           $target = substr( $target, 0, $hash );
                   } else {
                           $suffix = '';
                   }
                   # T9425
                   $target = trim( $target );
                   $contextPrefixedText = MediaWikiServices::getInstance()->
↪getTitleFormatter()->
                           getPrefixedText( $contextTitle );
                   # Look at the first character
                   if ( $target != '' && $target[0] === '/' ) {
                           # / at end means we don't want the slash to be shown
                           $m = [];
                           $trailingSlashes = preg_match_all( '%(/+)$%', $target, $m␣
↪);
                           if ( $trailingSlashes ) {
                                   $noslash = $target = substr( $target, 1, -strlen(
↪$m[0][0] ) );
                           } else {
                                   $noslash = substr( $target, 1 );
                           }

                           $ret = $contextPrefixedText . '/' . trim( $noslash ) .
↪$suffix;
                           if ( $text === '' ) {
                                   $text = $target . $suffix;
                           } # this might be changed for ugliness reasons
                   } else {
                           # check for .. subpage backlinks
                           $dotdotcount = 0;
                           $nodotdot = $target;
                           while ( strncmp( $nodotdot, "../", 3 ) == 0 ) {
                                   ++$dotdotcount;
                                   $nodotdot = substr( $nodotdot, 3 );
                           }
                           if ( $dotdotcount > 0 ) {
                                   $exploded = explode( '/', $contextPrefixedText );
                                   if ( count( $exploded ) > $dotdotcount ) { # not␣
↪allowed to go below top level page
                                           $ret = implode( '/', array_slice(
↪$exploded, 0, -$dotdotcount ) );
                                           # / at the end means don't show full path
                                           if ( substr( $nodotdot, -1, 1 ) === '/' )
↪{
                                                   $nodotdot = rtrim( $nodotdot, '/'␣
↪);
                                                   if ( $text === '' ) {
                                                           $text = $nodotdot .
↪$suffix;
                                                   }
                                           }
                                           $nodotdot = trim( $nodotdot );
                                           if ( $nodotdot != '' ) {
                                                   $ret .= '/' . $nodotdot;
                                           }
                                           $ret .= $suffix;
```

(continues on next page)

(continued from previous page)

```
                                    }
                            }
                    }
            }

            return $ret;
    }
```

10.2.127 Cast To Boolean

MediaWiki

Cast To Boolean, in includes/page/WikiPage.php:2274.

$options['changed'] and $options['created'] are documented and used as boolean. Yet, SiteStatsUpdate may require integers, for correct storage in the database, hence the type casting. `(int) (bool)` may be an alternative here.

```
$edits = $options['changed'] ? 1 : 0;
            $pages = $options['created'] ? 1 : 0;


            DeferredUpdates::addUpdate( SiteStatsUpdate::factory(
                    [ 'edits' => $edits, 'articles' => $good, 'pages' => $pages ]
            ) );
```

Dolibarr

Cast To Boolean, in htdocs/societe/class/societe.class.php:2777.

Several cases are built on the same pattern there. Each of the expression may be replaced by a cast to `(bool)`.

```
case 3:
                            $ret=(!$conf->global->SOCIETE_IDPROF3_UNIQUE?false:true);
                            break;
```

10.2.128 Failed Substr Comparison

Zurmo

Failed Substr Comparison, in app/protected/modules/zurmo/modules/SecurableModule.php:117.

filterAuditEvent compares a six char string with 'AUDIT_EVENT_' which contains 10 chars. This method returns only FALSE. Although it is used only once, the whole block that calls this method is now dead code.

```
private static function filterAuditEvent($s)
        {
            return substr($s, 0, 6) == 'AUDIT_EVENT_';
        }
```

MediaWiki

Failed Substr Comparison, in includes/media/DjVu.php:263.

$metadata contains data that may be in different formats. When it is a pure XML file, it is 'Old style'. The comment helps understanding that this is not the modern way to go : the Old Style is actually never called, due to a failing condition.

```
private function getUnserializedMetadata( File $file ) {
            $metadata = $file->getMetadata();
            if ( substr( $metadata, 0, 3 ) === '<?xml' ) {
                    // Old style. Not serialized but instead just a raw string of XML.
                    return $metadata;
            }
```

10.2.129 Use Positive Condition

SPIP

Use Positive Condition, in ecrire/inc/utils.php:925.

if (isset($time[$t])) { } else { } would put the important case in first place, and be more readable.

```
if (!isset($time[$t])) {
            $time[$t] = $a + $b;
    } else {
            $p = ($a + $b - $time[$t]) * 1000;
            unset($time[$t]);
#                   echo "'$p'";exit;
            if ($raw) {
                    return $p;
            }
            if ($p < 1000) {
                    $s = '';
            } else {
                    $s = sprintf("%d ", $x = floor($p / 1000));
                    $p -= ($x * 1000);
            }

            return $s . sprintf($s ? "%07.3f ms" : "%.3f ms", $p);
    }
```

ExpressionEngine

Use Positive Condition, in system/ee/EllisLab/Addons/forum/mod.forum_core.php:9138.

Let's be positive, and start processing the presence of $topic first. And let's call it empty(), not == ''.

```
if ($topic != '')
                                            {
                                                    $sql .= '('.substr($topic, 0, -3).
↪') OR ';
                                                    $sql .= '('.substr($tbody, 0, -3).
↪') ';
```

(continues on next page)

(continued from previous page)

```
                                            }
                                            else
                                            {
                                                    $sql = substr($sql, 0, -3);
                                            }
```

10.2.130 Don’t Echo Error

ChurchCRM

Don’t Echo Error, in wp-admin/includes/misc.php:74.

This is classic debugging code that should never reach production. mysqli_error() and mysqli_errno() provide valuable information is case of an error, and may be exploited by intruders.

```
if (mysqli_error($cnInfoCentral) != '') {
        echo gettext('An error occured: ').mysqli_errno($cnInfoCentral).'--'.mysqli_
↪error($cnInfoCentral);
    } else {
```

Phpdocumentor

Don’t Echo Error, in src/phpDocumentor/Plugin/Graphs/Writer/Graph.php:77.

Default development behavior : display the caught exception. Production behavior should not display that message, but log it for later review. Also, the return in the catch should be moved to the main code sequence.

```
public function processClass(ProjectDescriptor $project, Transformation
↪$transformation)
    {
        try {
            $this->checkIfGraphVizIsInstalled();
        } catch (\Exception $e) {
            echo $e->getMessage();

            return;
        }
```

10.2.131 Useless Casting

FuelCMS

Useless Casting, in fuel/codeigniter/core/URI.php:214.

substr() always returns a string, so there is no need to enforce this.

```
if (isset($_SERVER['SCRIPT_NAME'][0]))
            {
                    if (strpos($uri, $_SERVER['SCRIPT_NAME']) === 0)
                    {
```

(continues on next page)

(continued from previous page)

```
                            $uri = (string) substr($uri, strlen($_SERVER['SCRIPT_NAME
↪']));
                    }
                    elseif (strpos($uri, dirname($_SERVER['SCRIPT_NAME'])) === 0)
                    {
                            $uri = (string) substr($uri, strlen(dirname($_SERVER[
↪'SCRIPT_NAME'])));
                    }
            }
```

ThinkPHP

Useless Casting, in ThinkPHP/Library/Think/Db/Driver/Sqlsrv.class.php:67.

A comparison always returns a boolean, except for the spaceship operator.

```
foreach ($result as $key => $val) {
                $info[$val['column_name']] = array(
                    'name'    => $val['column_name'],
                    'type'    => $val['data_type'],
                    'notnull' => (bool) ('' === $val['is_nullable']), // not null is␣
↪empty, null is yes
                    'default' => $val['column_default'],
                    'primary' => false,
                    'autoinc' => false,
                );
            }
```

10.2.132 No isset() With empty()

XOOPS

No isset() With empty(), in htdocs/class/tree.php:297.

Too much vlaidation

```
isset($this->tree[$key]['child']) && !empty($this->tree[$key]['child']);
```

10.2.133 Useless Check

Magento

Useless Check, in wp-admin/includes/misc.php:74.

This code assumes that $delete is an array, then checks if it empty. Foreach will take care of the empty check.

```
if (!empty($delete)) {
            foreach ($delete as $categoryId) {
                $where = array(
                    'product_id = ?'  => (int)$object->getId(),
```

(continues on next page)

(continued from previous page)

```
                    'category_id = ?' => (int)$categoryId,
                );

                $write->delete($this->_productCategoryTable, $where);
            }
        }
```

Phinx

Useless Check, in src/Phinx/Migration/Manager.php:828.

If $dependencies is not empty, foreach() skips the loops.

```
private function getSeedDependenciesInstances(AbstractSeed $seed)
    {
        $dependenciesInstances = [];
        $dependencies = $seed->getDependencies();
        if (!empty($dependencies)) {
            foreach ($dependencies as $dependency) {
                foreach ($this->seeds as $seed) {
                    if (get_class($seed) === $dependency) {
                        $dependenciesInstances[get_class($seed)] = $seed;
                    }
                }
            }
        }

        return $dependenciesInstances;
    }
```

10.2.134 Bail Out Early

OpenEMR

Bail Out Early, in interface/modules/zend_modules/module/Carecoordination/src/Carecoordination/Controller/EncounterccdadispatchC

This is a typical example of a function mostly controlled by one condition. It could be rewrite as 'if($validResult != 'existingpatient')' then return. The 'else' clause is not used anymore, and the whole block of code is now the main sequence of the method.

```
public function ccdaFetching($parameterArray = array())
    {
        $validResult = $this->getEncounterccdadispatchTable()->valid(
↪$parameterArray[0]);
        // validate credentials
        if ($validResult == 'existingpatient') {
/// Long bloc of code
        } else {
            return '<?xml version=1.0 encoding=UTF-8?>
                    <!-- Edited by XMLSpy -->
                    <note>
```

(continues on next page)

(continued from previous page)

```
                            <heading>Authetication Failure</heading>
                            <body></body>
                    </note>
                    ';
        }
```

opencfp

Bail Out Early, in chair/assign_auto_reviewers_weighted_topic_match.inc:105.

This long example illustrates two aspects : first, the shortcut to the end of the method may be the 'then' clause, not necessarily the 'else'. '!in_array($pid.'-'.$rid, $conflictAR)' leads to return, and the 'else' should be removed, while keeping its content. Secondly, we can see 3 conditions that all lead to a premature end to the method. After refactoring all of them, the method would end up with 1 level of indentation, instead of 3.

```
function oc_inConflict(&$conflictAR, $pid, $rid=null) {
    if ($rid == null) {
            $rid = $_SESSION[OCC_SESSION_VAR_NAME]['acreviewerid'];
    }
    if (!in_array($pid.'-'.$rid, $conflictAR)) {
            return false; // not in conflict
    } else {
            $tempr = ocsql_query("SELECT COUNT(*) AS `count` FROM `" . OCC_TABLE_
↪PAPERREVIEWER . "` WHERE `paperid`='" . safeSQLstr($pid) . "' AND `reviewerid`='" .␣
↪safeSQLstr($rid) . "'");
            if ((ocsql_num_rows($tempr) == 1)
                    && ($templ = ocsql_fetch_assoc($tempr))
                    && ($templ['count'] == 1)
            ) {
                    return false; // assigned as reviewer
            } else {
                    $tempr = ocsql_query("SELECT COUNT(*) AS `count` FROM `" . OCC_
↪TABLE_PAPERADVOCATE . "` WHERE `paperid`='" . safeSQLstr($pid) . "' AND␣
↪`advocateid`='" . safeSQLstr($rid) . "'");
                    if ((ocsql_num_rows($tempr) == 1)
                            && ($templ = ocsql_fetch_assoc($tempr))
                            && ($templ['count'] == 1)
                    ) {
                            return false; // assigned as advocate
                    }
            }
    }
    return true;
}
```

10.2.135 Too Many Local Variables

HuMo-Gen

Too Many Local Variables, in relations.php:813.

15 local variables pieces of code are hard to find in a compact form. This function shows one classic trait of such issue : a large ifthen is at the core of the function, and each time, it collects some values and build a larger string. This should probably be split between different methods in a class.

```
function calculate_nephews($generX) { // handed generations x is removed from common␣
↪ancestor
global $db_functions, $reltext, $sexe, $sexe2, $language, $spantext, $selected_
↪language, $foundX_nr, $rel_arrayX, $rel_arrayspouseX, $spouse;
global $reltext_nor, $reltext_nor2; // for Norwegian and Danish

    if($selected_language=="es"){
            if($sexe=="m") { $neph=__('nephew'); $span_postfix="o "; $grson='nieto'; }
            else { $neph=__('niece'); $span_postfix="a "; $grson='nieta'; }
            //$gendiff = abs($generX - $generY); // FOUT
            $gendiff = abs($generX - $generY) - 1;
            $gennr=$gendiff-1;
            $degree=$grson." ".$gennr.$span_postfix;
            if($gendiff ==1) { $reltext=$neph.__(' of ');}
            elseif($gendiff > 1 AND $gendiff < 27) {
                    spanish_degrees($gendiff,$grson);
                    $reltext=$neph." ".$spantext.__(' of ');
            }
            else { $reltext=$neph." ".$degree; }
    } elseif ($selected_language==he){
            if($sexe=='m') { $nephniece = __('nephew'); }
///............
```

10.2.136 Illegal Name For Method

PrestaShop

Illegal Name For Method, in admin-dev/ajaxfilemanager/inc/class.pagination.php:200.

__getBaseUrl and __setBaseUrl shouldn't be named like that.

```
/**
     * get base url for pagination links aftr excluded those key
     * identified on excluded query strings
     *
     */
    function __getBaseUrl()
    {

            if(empty($this->baseUrl))
            {

                    $this->__setBaseUrl();
            }
            return $this->baseUrl;
    }
```

Magento

Illegal Name For Method, in app/code/core/Mage/Core/Block/Abstract.php:1139.

public method, called '__'. Example : $this->__();

```
public function __()
    {
        $args = func_get_args();
        $expr = new Mage_Core_Model_Translate_Expr(array_shift($args), $this->
↪getModuleName());
        array_unshift($args, $expr);
        return $this->_getApp()->getTranslator()->translate($args);
    }
```

10.2.137 Long Arguments

Cleverstyle

Long Arguments, in core/drivers/DB/MySQLi.php:40.

This query is not complex, but its length tend to push the end out of the view in the IDE. It could be rewritten as a variable, on the previous line, with some formatting. The same formatting would help without the variable too, yet, mixing the SQL syntax with the PHP methodcall adds a layer of confusion.

```
$this->instance->query("SET SESSION sql_mode='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,
↪NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_
↪ENGINE_SUBSTITUTION'")
```

Contao

Long Arguments, in core-bundle/src/Resources/contao/widgets/CheckBoxWizard.php:145.

This one-liner includes 9 members and 6 variables : some are formatted by sprintf, some are directly concatenated in the string. Breaking this into two lines improves readbility and code review.

```
sprintf('<span><input type="checkbox" name="%s" id="opt_%s" class="tl_checkbox" value=
↪"%s"%s%s onfocus="Backend.getScrollOffset()"> %s<label for="opt_%s">%s</label></
↪span>', $this->strName . ($this->multiple ? '[]' : ''), $this->strId . '_' . $i, (
↪$this->multiple ? \StringUtil::specialchars($arrOption['value']) : 1), (((\is_array(
↪$this->varValue) && \in_array($arrOption['value'], $this->varValue)) || $this->
↪varValue == $arrOption['value']) ? ' checked="checked"' : ''), $this->
↪getAttributes( ), $strButtons, $this->strId . '_' . $i, $arrOption['label'])
```

10.2.138 No Boolean As Default

OpenConf

No Boolean As Default, in openconf/include.php:1264.

Why do we need a *chair* when printing a cell's file ?

```
function oc_printFileCells(&$sub, $chair = false) { /**/ }
```

10.2.139 Property Used In One Method Only

Contao

Property Used In One Method Only, in calendar-bundle/src/Resources/contao/modules/ModuleEventlist.php:38.

Date is protected property. It is used only in the compile() method, and it is not used by the parent class. As such, it may be turned into a local variable.

```
class ModuleEventlist extends Events
{

    /**
     * Current date object
     * @var Date
     */
    protected $Date;

// Date is used in function compile() only
```

10.2.140 __DIR__ Then Slash

Traq

__DIR__ Then Slash, in src/Kernel.php:60.

When executed in a path '/a/b/c', this code will require '/a../../vendor/autoload.php.

```
static::$loader = require __DIR__.'../../vendor/autoload.php';
```

10.2.141 No Need For Else

Thelia

No Need For Else, in core/lib/Thelia/Core/Template/Loop/Address.php:92.

After checking that $currentCustomer is null, the method returns. The block with Else may be removed and its code may be moved one level up.

```
if ($customer === 'current') {
            $currentCustomer = $this->securityContext->getCustomerUser();
            if ($currentCustomer === null) {
                return null;
            } else {
                $search->filterByCustomerId($currentCustomer->getId(),␣
↪Criteria::EQUAL);
            }
        } else {
            $search->filterByCustomerId($customer, Criteria::EQUAL);
        }
```

ThinkPHP

No Need For Else, in projects/thinkphp/code//ThinkPHP/Library/Org/Util/Rbac.class.php:187.

This code has both good and bad example. Good : no use of else, after $_SESSION[$accessGuid] check. Issue : else usage after usage of !isset($accessList[strtoupper($appName)][strtoupper(CONTROLLER_NAME)][strtoupper(ACTION_NAME)])

```
if (empty($_SESSION[C('ADMIN_AUTH_KEY')])) {
                if (C('USER_AUTH_TYPE') == 2) {
                    //
                    //
                    $accessList = self::getAccessList($_SESSION[C('USER_AUTH_KEY')]);
                } else {
                    //
                    if ($_SESSION[$accessGuid]) {
                        return true;
                    }
                    //
                    $accessList = $_SESSION['_ACCESS_LIST'];
                }
                //
                if (!isset($accessList[strtoupper($appName)][strtoupper(CONTROLLER_
↪NAME)][strtoupper(ACTION_NAME)])) {
                    $_SESSION[$accessGuid] = false;
                    return false;
                } else {
                    $_SESSION[$accessGuid] = true;
                }
```

10.2.142 Strange Name For Variables

FuelCMS

Strange Name For Variables, in fuel/modules/fuel/libraries/parser/dwoo/Dwoo/Adapters/CakePHP/dwoo.php:86.

Three _ is quite a lot for variables. Would they not be parameters but global variables, that would still be quite a lot.

```
public function _render($___viewFn, $___data_for_view, $___play_safe = true,
↪$loadHelpers = true) {
    /**/
}
```

PhpIPAM

Strange Name For Variables, in app/admin/sections/edit-result.php:56.

$sss is the end-result of a progression, from $subsections (3s) to $ss to $sss. Although it is understandable from the code, a fuller name, like $subsection_subnet or $one_subsection_subnet would make this more readable.

```
//fetch subsection subnets
            foreach($subsections as $ss) {
                    $subsection_subnets = $Subnets->fetch_section_subnets($ss->id); //
↪fetch all subnets in subsection
```

(continues on next page)

(continued from previous page)

```
                    if(sizeof($subsection_subnets)>0) {
                            foreach($subsection_subnets as $sss) {
                                    $out[] = $sss;
                            }
                    }
                    $num_subnets = $num_subnets + sizeof($subsection_subnets);
                    //count all addresses that will be deleted!
                    $ipcnt = $Addresses->count_addresses_in_multiple_subnets($out);
            }
```

10.2.143 Check All Types

Zend-Config

Check All Types, in src/Writer/Ini.php:122.

$value must be an array or a string here.

```
foreach ($config as $key => $value) {
            $group = array_merge($parents, [$key]);

            if (is_array($value)) {
                $iniString .= $this->addBranch($value, $group);
            } else {
                $iniString .= implode($this->nestSeparator, $group)
                           .  ' = '
                           .  $this->prepareValue($value)
                           .  \n;
            }
        }
```

Vanilla

Check All Types, in library/core/class.form.php:2488.

When $this->_FormValues is not null, then it is an array or an object, as it may be used immediately with foreach(). A check with is_array() would be a stronger option here.

```
public function formDataSet() {
        if (is_null($this->_FormValues)) {
            $this->formValues();
        }

        $result = [[]];
        foreach ($this->_FormValues as $key => $value) {
```

10.2.144 Missing Cases In Switch

Tikiwiki

Missing Cases In Switch, in lib/articles/artlib.php:1075.

This switch handles 3 cases, plus the default for all others. There are other switch structures which also handle the ‘’ case. There may be a missing case here. In particular, projects/tikiwiki/code//article_image.php host another switch with the same case, plus another ‘topic’ case.

```
switch ($image_type) {
                    case 'article':
                            $image_cache_prefix = 'article';
                            break;
                    case 'submission':
                            $image_cache_prefix = 'article_submission';
                            break;
                    case 'preview':
                            $image_cache_prefix = 'article_preview';
                            break;
                    default:
                            return false;
            }
```

10.2.145 Repeated Regex

Vanilla

Repeated Regex, in library/core/class.pluginmanager.php:1200.

This regex is actually repeated 4 times across the Vanilla database, including this variation : ‘#^(https?:)?//#i’.

```
'`^https?://`'
```

Tikiwiki

Repeated Regex, in tiki-login.php:369.

This regex is use twice, identically, in the same file, with a few line of distance. It may be federated at the file level.

```
preg_match('/(tiki-register|tiki-login_validate|tiki-login_scr)\.php/', $url)
```

10.2.146 No Class In Global

Dolphin

No Class In Global, in Dolphin-v.7.3.5/inc/classes/BxDolXml.php:10.

This class should be put away in a ‘dolphin’ or ‘boonex’ namespace.

```
class BxDolXml {
    /* class BxDolXML code */
}
```

10.2.147 Could Use str_repeat()

Zencart

Could Use str_repeat(), in includes/functions/functions_general.php:1234.

That’s a 45 repeat of &nbsp;

```
if ( (!zen_browser_detect('MSIE')) && (zen_browser_detect('Mozilla/4')) ) {
      for ($i=0; $i<45; $i++) $pre .= '&nbsp;';
    }
```

10.2.148 Suspicious Comparison

PhpIPAM

Suspicious Comparison, in app/tools/vrf/index.php:110.

if $subnet[‘description’] is a string, the comparison with 0 turn it into a boolean. false’s length is 0, and true length is 1. PHP saves the day.

```
$subnet['description'] = strlen($subnet['description']==0) ? "/" : $subnet[
↪'description'];
```

ExpressionEngine

Suspicious Comparison, in ExpressionEngine_Core2.9.2/system/expressionengine/libraries/simplepie/SimplePie/Misc.php:1925.

If trim($attribs[‘’][‘mode’]) === ‘base64’, then it is set to lowercase (although it is already), and added to the && logical test. If it is ‘BASE64’, this fails.

```
if (isset($attribs['']['mode']) && strtolower(trim($attribs['']['mode']) === 'base64
↪'))
```

10.2.149 Strings With Strange Space

OpenEMR

Strings With Strange Space, in library/globals.inc.php:3270.

The name of the contry contains both an unsecable space (the first, after Tonga), and a normal space (between Tonga and Islands). Translations are stored in a database, which preserves the unbreakable spaces. This also means that fixing the translation must be applied to every piece of data at the same time. The xl() function, which handles the translations, is also a good place to clean the spaces before searching for the right translation.

```
'to' => xl('Tonga (Tonga Islands)'),
```

Thelia

Strings With Strange Space, in templates/backOffice/default/I18n/fr_FR.php:647.

This is another example with a translation sentence. Here, the unbreakable space is before the question mark : this is a typography rule, that is common to many language. This would be a false positive, unless typography is handled by another part of the software.

```
'Mot de passe oublié ?'
```

10.2.150 No Empty Regex

Tikiwiki

No Empty Regex, in lib/sheet/excel/writer/worksheet.php:1925.

The initial 's' seems to be too much. May be a typo ?

```
// Strip URL type
        $url = preg_replace('s[^internal:]', '', $url);
```

10.2.151 Randomly Sorted Arrays

Contao

Randomly Sorted Arrays, in system/modules/core/dca/tl_module.php:259.

The array array('maxlength', 'decodeEntities', 'tl_class') is configured multiple times in this file. Most of them is in the second form, but some are in the first form. (Multiple occurrences in this file).

```
array('maxlength' => 255, 'decodeEntities' => true, 'tl_class' => 'w50') // Line 246
array('decodeEntities' => true, 'maxlength' => 255, 'tl_class' => 'w50'); // ligne 378
```

Vanilla

Randomly Sorted Arrays, in applications/dashboard/models/class.activitymodel.php:308.

'Photo' moved from last to second. This array is used with a 'Join' key, and is the base for a SQL table JOIN. As such, order is important. If this is the case, it seems unusual that the order is not the same for a join using the same tables. If it is not the case, arrays may be reordered.

```
/* L 305 */        Gdn::userModel()->joinUsers(
            $result->resultArray(),
            ['ActivityUserID', 'RegardingUserID'],
            ['Join' => ['Name', 'Email', 'Gender', 'Photo']]
        );

// L 385
        Gdn::userModel()->joinUsers($result, ['ActivityUserID', 'RegardingUserID'], [
↪'Join' => ['Name', 'Photo', 'Email', 'Gender']]);
```

10.2.152 Only Variable Passed By Reference

Dolphin

Only Variable Passed By Reference, in administration/charts.json.php:89.

This is not possible, as array_slice() returns a new array, and not a reference. Minimally, the intermediate result must be saved in a variable, then popped. Actually, this code extracts the element at key 1 in the $aData array, although this also works with hash (non-numeric keys).

```
array_pop(array_slice($aData, 0, 1))
```

PhpIPAM

Only Variable Passed By Reference, in functions/classes/class.Thread.php:243.

This is sneaky bug : the assignation $status = 0 returns a value, and not a variable. This leads PHP to mistake the initialized 0 with the variable $status and fails. It is not possible to initialize variable AND use them as argument.

```
pcntl_waitpid($this->pid, $status = 0)
```

10.2.153 No Return Used

SPIP

No Return Used, in ecrire/inc/utils.php:1067.

job_queue_remove() is called as an administration order, and the result is not checked. It is considered as a fire-and-forget command.

```
function job_queue_remove($id_job) {
    include_spip('inc/queue');

    return queue_remove_job($id_job);
}
```

LiveZilla

No Return Used, in livezilla/_lib/trdp/Zend/Loader.php:114.

The loadFile method tries to load a file, aka as include. If the inclusion fails, a PHP error is emitted (an exception would do the same), and there is not error management. Hence, the ‘return true;’, which is not tested later. It may be dropped.

```
public static function loadFile($filename, $dirs = null, $once = false)
    {
// A lot of code to check and include files

        return true;
    }
```

10.2.154 Mixed Concat And Interpolation

SuiteCrm

Mixed Concat And Interpolation, in modules/AOW_Actions/actions/actionSendEmail.php:89.

How long did it take to spot the hidden $checked variable in this long concatenation ? Using a consistent method of interpolation would help readability here.

```
"<input type='checkbox' id='aow_actions_param[" . $line . "][individual_email]' name=
↪'aow_actions_param[" . $line . "][individual_email]' value='1' $checked></td>"
```

Edusoho

Mixed Concat And Interpolation, in src/AppBundle/Controller/Admin/SiteSettingController.php:168.

Calling a method from a property of an object is possible inside a string, though it is rare. Setting the method outside the string make it more readable.

```
"{$this->container->getParameter('topxia.upload.public_url_path')}/" . $parsed['path']
```

10.2.155 Too Many Injections

NextCloud

Too Many Injections, in lib/private/Share20/Manager.php:130.

Well documented Manager class. Quite a lot of injections though, it must take a long time to prepare it.

```
/**
     * Manager constructor.
     *
     * @param ILogger $logger
     * @param IConfig $config
     * @param ISecureRandom $secureRandom
     * @param IHasher $hasher
     * @param IMountManager $mountManager
     * @param IGroupManager $groupManager
     * @param IL10N $l
     * @param IFactory $l10nFactory
     * @param IProviderFactory $factory
     * @param IUserManager $userManager
     * @param IRootFolder $rootFolder
     * @param EventDispatcher $eventDispatcher
     * @param IMailer $mailer
     * @param IURLGenerator $urlGenerator
     * @param \OC_Defaults $defaults
     */
    public function __construct(
                    ILogger $logger,
                    IConfig $config,
                    ISecureRandom $secureRandom,
                    IHasher $hasher,
```

(continues on next page)

(continued from previous page)

```
                    IMountManager $mountManager,
                    IGroupManager $groupManager,
                    IL10N $l,
                    IFactory $l10nFactory,
                    IProviderFactory $factory,
                    IUserManager $userManager,
                    IRootFolder $rootFolder,
                    EventDispatcher $eventDispatcher,
                    IMailer $mailer,
                    IURLGenerator $urlGenerator,
                    \OC_Defaults $defaults
    ) {
            $this->logger = $logger;
            $this->config = $config;
            $this->secureRandom = $secureRandom;
            $this->hasher = $hasher;
            $this->mountManager = $mountManager;
            $this->groupManager = $groupManager;
            $this->l = $l;
            $this->l10nFactory = $l10nFactory;
            $this->factory = $factory;
            $this->userManager = $userManager;
            $this->rootFolder = $rootFolder;
            $this->eventDispatcher = $eventDispatcher;
            $this->sharingDisabledForUsersCache = new CappedMemoryCache();
            $this->legacyHooks = new LegacyHooks($this->eventDispatcher);
            $this->mailer = $mailer;
            $this->urlGenerator = $urlGenerator;
            $this->defaults = $defaults;
    }
```

Thelia

Too Many Injections, in core/lib/Thelia/Core/Event/Delivery/DeliveryPostageEvent.php:58.

Classic address class, with every details. May be even shorter than expected.

```
//class DeliveryPostageEvent extends ActionEvent
    public function __construct(
        DeliveryModuleInterface $module,
        Cart $cart,
        Address $address = null,
        Country $country = null,
        State $state = null
    ) {
        $this->module = $module;
        $this->cart = $cart;
        $this->address = $address;
        $this->country = $country;
        $this->state = $state;
    }
```

10.2.156 @ Operator

Phinx

@ Operator, in src/Phinx/Util/Util.php:239.

fopen() may be tested for existence, readability before using it. Although, it actually emits some errors on Windows, with network volumes.

```
$isReadable = @\fopen($filePath, 'r') !== false;

        if (!$filePath || !$isReadable) {
            throw new \Exception(sprintf(Cannot open file %s \n, $filename));
        }
```

PhpIPAM

@ Operator, in functions/classes/class.Log.php:322.

Variable and index existence should always be tested with isset() : it is faster than using `@`.

```
$_SESSION['ipamusername']
```

10.2.157 Avoid Optional Properties

ChurchCRM

Avoid Optional Properties, in src/ChurchCRM/BackupManager.php:401.

Backuptype is initialized with null, and yet, it isn't checked for any invalid valid values, in particular in switch() structures.

```
// BackupType is initialized with null
  class JobBase
  {
      /**
        *
        * @var BackupType
        */
      protected $BackupType;

// In the child class BackupJob, BackupType may be of any type
  class BackupJob extends JobBase
  {
      /**
       *
       * @param String $BaseName
       * @param BackupType $BackupType
       * @param Boolean $IncludeExtraneousFiles
       */
      public function __construct($BaseName, $BackupType, $IncludeExtraneousFiles,
↪$EncryptBackup, $BackupPassword)
      {
          $this->BackupType = $BackupType;
```

(continues on next page)

(continued from previous page)

```
// Later, Backtype is not checked with all values :
          try {
              $this->DecryptBackup();
              switch ($this->BackupType) {
              case BackupType::SQL:
                $this->RestoreSQLBackup($this->RestoreFile);
                break;
              case BackupType::GZSQL:
                $this->RestoreGZSQL();
                break;
              case BackupType::FullBackup:
                $this->RestoreFullBackup();
                break;
// Note  : no default case here
            }
```

Dolibarr

Avoid Optional Properties, in htdocs/product/stock/class/productlot.class.php:149.

$this->fk_product is tested for value 11 times while being used in this class. All detected situations were checking the presence of the property before usage.

```
class Productlot extends CommonObject
{
// more code
    /**
     * @var int ID
     */
    public $fk_product;

// Checked usage of fk_product
// line 341
            $sql .= ' fk_product = '.(isset($this->fk_product) ? $this->fk_product :
↪"null").',';
```

10.2.158 Mismatched Ternary Alternatives

phpadsnew

Mismatched Ternary Alternatives, in phpAdsNew-2.0/admin/lib-misc-stats.inc.php:219.

This is an unusual way to apply a condition. $bgcolor is '#FFFFFF' by default, and if $i % 2, then $bcolor is '#F6F6F6';. A more readable ternary option would be '$bgcolor = = $i % 2 ? "#FFFFFF" : "#F6F6F6";', and make a matched alternative branches.

```
$bgcolor = #FFFFFF;
    $i % 2 ? 0 : $bgcolor = #F6F6F6;
```

OpenEMR

Mismatched Ternary Alternatives, in portal/messaging/messages.php:132.

IS_DASHBOARD is defined as a boolean or a string. Later, it is tested as a boolean, and displayed as a integer, which will be cast to string by echo. Lots of transtyping are happening here.

```
// In two distinct if/then branch
l:29) define('IS_DASHBOARD', false);
l:41) define('IS_DASHBOARD', $_SESSION['authUser']);

l:132) echo IS_DASHBOARD ? IS_DASHBOARD : 0;
?>
```

10.2.159 Mismatched Default Arguments

SPIP

Mismatched Default Arguments, in ecrire/inc/lien.php:160.

generer_url_entite() takes $connect in, with a default value of empty string. Later, generer_url_entite() receives that value, but uses null as a default value. This forces the ternary test on $connect, to turn it into a null before shipping it to the next function, and having it processed accordingly.

```
// http://code.spip.net/@traiter_lien_implicite
function traiter_lien_implicite($ref, $texte = '', $pour = 'url', $connect = '') {

    // some code was edited here

    if (is_array($url)) {
            @list($type, $id) = $url;
            $url = generer_url_entite($id, $type, $args, $ancre, $connect ? $connect␣
↪: null);
    }
```

10.2.160 Mismatched Typehint

WordPress

Mismatched Typehint, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

10.2.161 Scalar Or Object Property

SugarCrm

Scalar Or Object Property, in SugarCE-Full-6.5.26/data/Link.php:54.

The _relationship property starts its life as a string, and becomes an object later.

```
class Link {

    /* Private variables.*/
    var $_log;
    var $_relationship_name; //relationship this attribute is tied to.
    var $_bean; //stores a copy of the bean.
    var $_relationship= '';

/// More code.....

// line 92
            $this->_relationship=new Relationship();
```

10.2.162 Assign With And

xataface

Assign With And, in Dataface/LanguageTool.php:265.

The usage of 'and' here is a workaround for PHP version that have no support for the coalesce. $autosubmit receives the value of $params['autosubmit'] only if the latter is set. Yet, with = having higher precedence over 'and', $autosubmit is mistaken with the existence of $params['autosubmit'] : its value is actually omitted.

```
$autosubmit = isset($params['autosubmit']) and $params['autosubmit'];
```

10.2.163 Logical To in_array

Zencart

Logical To in_array, in admin/users.php:32.

Long list of == are harder to read. Using an in_array() call gathers all the strings together, in an array. In turn, this helps readability and possibility, reusability by making that list an constant.

```
// if needed, check that a valid user id has been passed
if (($action == 'update' || $action == 'reset') && isset($_POST['user']))
{
  $user = $_POST['user'];
}
elseif (($action == 'edit' || $action == 'password' || $action == 'delete' || $action␣
↪== 'delete_confirm') && $_GET['user'])
{
  $user = $_GET['user'];
}
elseif(($action=='delete' || $action=='delete_confirm') && isset($_POST['user']))
{
  $user = $_POST['user'];
}
```

10.2.164 Pathinfo() Returns May Vary

NextCloud

Pathinfo() Returns May Vary, in lib/private/Preview/Office.php:56.

$absPath is build with the toTmpFile() method, which may return a boolean (false) in case of error. Error situations include the inability to create the temporary file.

```
$absPath = $fileview->toTmpFile($path);

// More code

                    list($dirname, , , $filename) = array_values(pathinfo($absPath));
                    $pngPreview = $dirname . '/' . $filename . '.png';
```

10.2.165 Multiple Type Variable

Typo3

Multiple Type Variable, in typo3/sysext/backend/Classes/Form/Element/InputDateTimeElement.php:270.

$fullElement is an array most of the time, but finally ends up being a string. Since the array is not the final state, it may be interesting to make it a class, which collects the various variables, and export the final string. Such class would be usefull in several places in this repository.

```
$fullElement = [];
            $fullElement[] = '<div class=checkbox t3js-form-field-eval-null-
↪placeholder-checkbox>';
            $fullElement[] =     '<label for= . $nullControlNameEscaped . >';
            $fullElement[] =         '<input type=hidden name= .
↪$nullControlNameEscaped .  value= . $fallbackValue .  />';
            $fullElement[] =         '<input type=checkbox name= .
↪$nullControlNameEscaped .  id= . $nullControlNameEscaped .  value=1' . $checked .
↪$disabled . ' />';
            $fullElement[] =         $overrideLabel;
            $fullElement[] =     '</label>';
            $fullElement[] = '</div>';
            $fullElement[] = '<div class=t3js-formengine-placeholder-placeholder>';
            $fullElement[] =    '<div class=form-control-wrap style=max-width: .
↪$width . px>';
            $fullElement[] =        '<input type=text class=form-control␣
↪disabled=disabled value= . $shortenedPlaceholder .  />';
            $fullElement[] =    '</div>';
            $fullElement[] = '</div>';
            $fullElement[] = '<div class=t3js-formengine-placeholder-formfield>';
            $fullElement[] =    $expansionHtml;
            $fullElement[] = '</div>';
            $fullElement = implode(LF, $fullElement);
```

Vanilla

Multiple Type Variable, in library/core/functions.general.php:1427.

Here, $value may be of different type. The if() structures merges all the incoming format into one standard type (int). This is actually the contrary of this analysis, and is a false positive.

```
if (is_array($value)) {
                        $value = count($value);
                    } elseif (stringEndsWith($field, 'UserID', true)) {
                        $value = 1;
                    }
```

10.2.166 Is Actually Zero

Dolibarr

Is Actually Zero, in htdocs/compta/ajaxpayment.php:99.

Here, the $amountToBreakDown is either $currentRemain or $result.

```
$amountToBreakdown = ($result - $currentRemain >= 0 ?

↪$currentRemain :                                                                   //␣
↪Remain can be fully paid

↪$currentRemain + ($result - $currentRemain));    // Remain can only partially be paid
```

SuiteCrm

Is Actually Zero, in modules/AOR_Charts/lib/pChart/class/pDraw.class.php:523.

$Xa may only amount to $iX2, though the expression looks weird.

```
if ( $X > $iX2 ) { $Xa = $X-($X-$iX2); $Ya = $iY1+($X-$iX2); } else { $Xa = $X; $Ya =
↪$iY1; }
```

10.2.167 Unconditional Break In Loop

LiveZilla

Unconditional Break In Loop, in wp-admin/includes/misc.php:74.

Only one row is read from the DBManager, and the rest is ignored. The result has no more than one result, basedd on the *LIMIT 1* clause in the SQL. The while loop may be removed.

```
$result = DBManager::Execute(true, "SELECT * FROM `" . DB_PREFIX . DATABASE_STATS_
↪AGGS . "` WHERE `month`>0 AND ((`year`='" . DBManager::RealEscape(date("Y")) . "'␣
↪AND `month`<'" . DBManager::RealEscape(date("n")) . "') OR (`year`<'" .␣
↪DBManager::RealEscape(date("Y")) . "')) AND (`aggregated`=0 OR `aggregated`>" .␣
↪(time() - 300) . ") AND `day`=0 ORDER BY `year` ASC,`month` ASC LIMIT 1;");
        if ($result)
            while ($row = DBManager::FetchArray($result)) {
                if (empty($row["aggregated"])) {
                    DBManager::Execute(true, "UPDATE `" . DB_PREFIX . DATABASE_STATS_
↪AGGS . "` SET `aggregated`=" . time() . " WHERE `year`=" . $row["year"] . " AND␣
↪`month`=" . $row["month"] . " AND `day`=0 LIMIT 1;");
                    $this->AggregateMonth($row["year"], $row["month"]);
```

(continues on next page)

(continued from previous page)

```
                }
                return false;
            }
```

MediaWiki

Unconditional Break In Loop, in includes/htmlform/HTMLFormField.php:138.

The final break is useless : the execution has already reached the end of the loop.

```
for ( $i = count( $thisKeys ) - 1; $i >= 0; $i-- ) {
                    $keys = array_merge( array_slice( $thisKeys, 0, $i ), $nameKeys );
                    $data = $alldata;
                    foreach ( $keys as $key ) {
                            if ( !is_array( $data ) || !array_key_exists( $key, $data␣
↪) ) {
                                    continue 2;
                            }
                            $data = $data[$key];
                    }
                    $testValue = (string)$data;
                    break;
            }
```

10.2.168 Could Be Else

SugarCrm

Could Be Else, in SugarCE-Full-6.5.26/modules/Emails/ListViewGroup.php:79.

The first condition makes different checks if ‘query’ is in $_REQUEST or not. The second only applies to $_REQUEST[‘query’], as there is no else. There is also no visible sign that the first condition may change $_REQUEST or not

```
if(!isset($_REQUEST['query'])){
    //_pp('loading: '.$currentModule.'Group');
    //_pp($current_user->user_preferences[$currentModule.'GroupQ']);
    $storeQuery->loadQuery($currentModule.'Group');
    $storeQuery->populateRequest();
} else {
    //_pp($current_user->user_preferences[$currentModule.'GroupQ']);
    //_pp('saving: '.$currentModule.'Group');
    $storeQuery->saveFromGet($currentModule.'Group');
}

if(isset($_REQUEST['query'])) {
    // we have a query
    if(isset($_REQUEST['email_type']))                          $email_type = $_
↪REQUEST['email_type'];
    if(isset($_REQUEST['assigned_to']))                         $assigned_to = $_
↪REQUEST['assigned_to'];
    if(isset($_REQUEST['status']))                              $status = $_
↪REQUEST['status'];
```

(continues on next page)

(continued from previous page)

```
    // More code
}
```

OpenEMR

Could Be Else, in library/log.inc:653.

Those two if structure may definitely merged into one single instruction.

```
$success = 1;
    $checksum = ;
    if ($outcome === false) {
        $success = 0;
    }

    if ($outcome !== false) {
        // Should use the $statement rather than the processed
        // variables, which includes the binded stuff. If do
        // indeed need the binded values, then will need
        // to include this as a separate array.

        //error_log(STATEMENT: .$statement,0);
        //error_log(BINDS: .$processed_binds,0);
        $checksum = sql_checksum_of_modified_row($statement);
        //error_log(CHECKSUM: .$checksum,0);
    }
```

10.2.169 Next Month Trap

Contao

Next Month Trap, in system/modules/calendar/classes/Events.php:515.

This code is wrong on August 29,th 30th and 31rst : 6 months before is caculated here as February 31rst, so march 2. Of course, this depends on the leap years.

```
case 'past_180':
                            return array(strtotime('-6 months'), time(), $GLOBALS['TL_
↪LANG']['MSC']['cal_empty']);
```

Edusoho

Next Month Trap, in src/AppBundle/Controller/Admin/AnalysisController.php:1426.

The last month is wrong 8 times a year : on 31rst, and by the end of March.

```
'lastMonthStart' => date('Y-m-d', strtotime(date('Y-m', strtotime('-1 month')))),
            'lastMonthEnd' => date('Y-m-d', strtotime(date('Y-m', time())) - 24 *␣
↪3600),
            'lastThreeMonthsStart' => date('Y-m-d', strtotime(date('Y-m', strtotime('-
↪2 month')))),
```

(continues on next page)

(continued from previous page)

10.2.170 Printf Number Of Arguments

PhpIPAM

Printf Number Of Arguments, in functions/classes/class.Common.php:1174.

16 will not be displayed.

```
sprintf('%032s', gmp_strval(gmp_init($ipv6long, 10), 16);
```

10.2.171 Don't Send $this In Constructor

Woocommerce

Don't Send $this In Constructor, in includes/class-wc-cart.php:107.

WC_Cart_Session and WC_Cart_Fees receives $this, the current object, at a moment where it is not consistent : for example, tax_display_cart hasn't been set yet. Although it may be unexpected to have an object called WC_Cart being called by the session or the fees, this is still a temporary inconsistence.

```
/**
     * Constructor for the cart class. Loads options and hooks in the init method.
     */
    public function __construct() {
            $this->session          = new WC_Cart_Session( $this );
            $this->fees_api         = new WC_Cart_Fees( $this );
            $this->tax_display_cart = $this->is_tax_displayed();

            // Register hooks for the objects.
            $this->session->init();
```

Contao

Don't Send $this In Constructor, in system/modules/core/library/Contao/Model.php:110.

$this is send to $objRegistry. $objRegistry is obtained with a factory, ModelRegistry::getInstance(). It is probably fully prepared at that point. Yet, $objRegistry is called and used to fill $this properties with full values. At some point, $objRegistry return values without having a handle on a fully designed object.

```
/**
     * Load the relations and optionally process a result set
     *
     * @param \Database\Result $objResult An optional database result
     */
    public function __construct(\Database\Result $objResult=null)
    {
        // Some code was removed
                    $objRegistry = \Model\Registry::getInstance();
```

(continues on next page)

(continued from previous page)

```
                    $this->setRow($arrData); // see #5439
                    $objRegistry->register($this);

        // More code below
        // $this-> are set
        // $objRegistry is called
    }
```

10.2.172 Parent First

shopware

Parent First, in wp-admin/includes/misc.php:74.

Here, the parent is called last. Givent that $title is defined in the same class, it seems that $name may be defined in the BaseContainer class. In fact, it is not, and BasecContainer and FieldSet are fairly independant classes. Thus, the parent::__construct call could be first here, though more as a coding convention.

```
/**
 * Class FieldSet
 */
class FieldSet extends BaseContainer
{
    /**
     * @var string
     */
    protected $title;

    /**
     * @param string $name
     * @param string $title
     */
    public function __construct($name, $title)
    {
        $this->title = $title;
        $this->name = $name;
        parent::__construct();
    }
```

PrestaShop

Parent First, in controllers/admin/AdminPatternsController.php:30.

A good number of properties are set in the current object even before the parent AdminController(Core) is called. ‘table’ and ‘lang’ acts as default values for the parent class, as it (the parent class) would set them to another default value. Many properties are used, but not defined in the current class, nor its parent. This approach prevents the constructor from requesting too many arguments. Yet, as such, it is difficult to follow which of the initial values are transmitted via protected/public properties rather than using the __construct() call.

```
class AdminPatternsControllerCore extends AdminController
{
    public $name = 'patterns';

    public function __construct()
    {
        $this->bootstrap = true;
        $this->show_toolbar = false;
        $this->context = Context::getContext();

        parent::__construct();
    }
```

10.2.173 Invalid Regex

SugarCrm

Invalid Regex, in SugarCE-Full-6.5.26/include/utils/file_utils.php:513.

This yields an error at execution time : “Compilation failed: invalid range in character class at offset 4 “.

```
preg_replace('/[^\w-._]+/i', '', $name)
```

10.2.174 Use Named Boolean In Argument Definition

phpMyAdmin

Use Named Boolean In Argument Definition, in /libraries/classes/Util.php:1929.

$request is an option to *checkParameters*, although it is not visibile with is its actual role.

```
public static function checkParameters($params, $request = false) {
    /**/
}
```

Cleverstyle

Use Named Boolean In Argument Definition, in /core/classes/Response.php:129.

$httponly is an option to *cookie*, and true/false makes it readable. There may be other situations, like fallback, or forcedd usage, so the boolean may be misleading. Note also the *$expire = 0*, which may be a date, or a special value. We need to read the documentation to understand this.

```
public function cookie($name, $value, $expire = 0, $httponly = false) { /**/ }        {
    /**/
}
```

10.2.175 Never Used Parameter

Piwigo

Never Used Parameter, in include/functions_html.inc.php:329.

$alternate_url is never explicitely passed to bad_request() : this doesn't show in this extract. It could be dropped from this code.

```
function bad_request($msg, $alternate_url=null)
{
  set_status_header(400);
  if ($alternate_url==null)
    $alternate_url = make_index_url();
  redirect_html( $alternate_url,
    '<div style="text-align:left; margin-left:5em;margin-bottom:5em;">
<h1 style="text-align:left; font-size:36px;">'.l10n('Bad request').'</h1><br>'
.$msg.'</div>',
    5 );
}
```

10.2.176 Identical On Both Sides

phpMyAdmin

Identical On Both Sides, in libraries/classes/DatabaseInterface.php:323.

This code looks like `($options & DatabaseInterface::QUERY_STORE) == DatabaseInterface::QUERY_STORE`, which would make sense. But PHP precedence is actually executing `$options & (DatabaseInterface::QUERY_STORE == DatabaseInterface::QUERY_STORE)`, which then doesn't depends on QUERY_STORE but only on $options.

```
if ($options & DatabaseInterface::QUERY_STORE == DatabaseInterface::QUERY_STORE) {
    $tmp = $this->_extension->realQuery('
        SHOW COUNT(*) WARNINGS', $this->_links[$link], DatabaseInterface::QUERY_STORE
    );
    $warnings = $this->fetchRow($tmp);
} else {
    $warnings = 0;
}
```

HuMo-Gen

Identical On Both Sides, in include/person_cls.php:73.

In that long logical expression, $personDb->pers_cal_date is tested twice

```
// *** Filter person's WITHOUT any date's ***
                    if ($user[group_filter_date]=='j'){
                            if ($personDb->pers_birth_date=='' AND $personDb->pers_
↪bapt_date==''
                            AND $personDb->pers_death_date=='' AND $personDb->pers_
↪buried_date==''
```

(continues on next page)

(continued from previous page)

```
                            AND $personDb->pers_cal_date=='' AND $personDb->pers_cal_
↪date==''
                            ){
                                    $privacy_person='';
                            }
                    }
```

10.2.177 No Reference For Ternary

phpadsnew

No Reference For Ternary, in lib/OA/Admin/Menu/Section.php334:334.

The reference should be removed from the function definition. Either this method returns null, which is never a reference, or it returns $this, which is always a reference, or the results of a methodcall. The latter may or may not be a reference, but the Ternary operator will drop it and return by value.

```
function &getParentOrSelf($type)
    {
        if ($this->type == $type) {
            return $this;
        }
        else {
            return $this->parentSection != null ? $this->parentSection->
↪getParentOrSelf($type) : null;
        }
    }
```

10.2.178 Unused Inherited Variable In Closure

shopware

Unused Inherited Variable In Closure, in recovery/update/src/app.php:129.

In the first closuree, $containere is used as the root for the method calls, but $app is not used. It may be dropped. In fact, some of the following calls to $app->map() only request one inherited, $container.

```
$app->map('/applyMigrations', function () use ($app, $container) {
    $container->get('controller.batch')->applyMigrations();
})->via('GET', 'POST')->name('applyMigrations');

$app->map('/importSnippets', function () use ($container) {
    $container->get('controller.batch')->importSnippets();
})->via('GET', 'POST')->name('importSnippets');
```

Mautic

Unused Inherited Variable In Closure, in MauticCrmBundle/Tests/Integration/SalesforceIntegrationTest.php:1202.

$max is relayed to getLeadsToCreate(), while $restart is omitted. It may be dropped, along with its reference.

```
function () use (&$restart, $max) {
                    $args = func_get_args();

                    if (false === $args[2]) {
                        return $max;
                    }

                    $createLeads = $this->getLeadsToCreate($args[2], $max);

                    // determine whether to return a count or records
                    if (false === $args[2]) {
                        return count($createLeads);
                    }

                    return $createLeads;
                }
```

10.2.179 Useless Referenced Argument

Woocommerce

Useless Referenced Argument, in includes/data-stores/class-wc-product-variation-data-store-cpt.php:414.

$product is defined with a reference in the method signature, but it is also used as an object with a dynamical property. As such, the reference in the argument definition is too much.

```
public function update_post_meta( &$product, $force = false ) {
            $meta_key_to_props = array(
                    '_variation_description' => 'description',
            );

            $props_to_update = $force ? $meta_key_to_props : $this->get_props_to_
↪update( $product, $meta_key_to_props );

            foreach ( $props_to_update as $meta_key => $prop ) {
                                    $value   = $product->{get_$prop}( 'edit' );
                                    $updated = update_post_meta( $product->get_id(),
↪$meta_key, $value );
                    if ( $updated ) {
                            $this->updated_props[] = $prop;
                    }
            }

            parent::update_post_meta( $product, $force );
```

Magento

Useless Referenced Argument, in setup/src/Magento/Setup/Module/Di/Compiler/Config/Chain/PreferencesResolving.php:63.

$value is defined with a reference. In the following code, it is only read and never written : for index search, or by itself. In fact, $preferences is also only read, and never written. As such, both could be removed.

```
private function resolvePreferenceRecursive(&$value, &$preferences)
    {
        return isset($preferences[$value])
            ? $this->resolvePreferenceRecursive($preferences[$value], $preferences)
            : $value;
    }
```

10.2.180 Useless Catch

Zurmo

Useless Catch, in app/protected/modules/workflows/forms/attributes/ExplicitReadWriteModelPermissionsWorkflowActionAttributeFor

Catch the exception, then return. At least, the comment is honest.

```
try
                {
                    $group = Group::getById((int)$this->type);
                    $explicitReadWriteModelPermissions->addReadWritePermitable(
↪$group);
                }
                catch (NotFoundException $e)
                {
                    //todo: handle exception better
                    return;
                }
```

PrestaShop

Useless Catch, in src/Core/Addon/Module/ModuleManagerBuilder.php:170.

Here, the catch clause will intercept a IO problem while writing element on the disk, and will return false. Since this is a constructor, the returned value will be ignored and the object will be left in a wrong state, since it was not totally inited.

```
private function __construct()
    {
    // More code......
            try {
                $filesystem = new Filesystem();
                $filesystem->dumpFile($phpConfigFile, '<?php return ' . var_export(
↪$config, true) . ';' . \n);
            } catch (IOException $e) {
                return false;
            }
        }
```

10.2.181 Test Then Cast

Dolphin

Test Then Cast, in wp-admin/includes/misc.php:74.

$aLimits['per_page'] is tested for existence and not false. Later, it is cast from string to int : yet, a '0.1' string value would pass the test, and end up filling $aLimits['per_page'] with 0.

```
if (isset($aLimits['per_page']) && $aLimits['per_page'] !== false)
            $this->aCurrent['paginate']['perPage'] = (int)$aLimits['per_page'];
```

SuiteCrm

Test Then Cast, in modules/jjwg_Maps/controller.php:1035.

$marker['lat'] is compared to the string '0', which actually transtype it to integer, then it is cast to string for map_marker_data_points() needs and finally, it is cast to float, in case of a correction. It would be safer to test it in its string type, since floats are not used as array indices.

```
if ($marker['lat'] != '0' && $marker['lng'] != '0') {

            // Check to see if marker point already exists and apply offset if needed
            // This often occurs when an address is only defined by city, state, zip.
            $i = 0;
            while (isset($this->map_marker_data_points[(string) $marker['lat
↪']][(string) $marker['lng']]) &&
            $i < $this->settings['map_markers_limit']) {
                $marker['lat'] = (float) $marker['lat'] + (float) $this->settings[
↪'map_duplicate_marker_adjustment'];
                $marker['lng'] = (float) $marker['lng'] + (float) $this->settings[
↪'map_duplicate_marker_adjustment'];
                $i++;
            }
```

10.2.182 Property Could Be Local

Mautic

Property Could Be Local, in app/bundles/EmailBundle/Model/SendEmailToContact.php:47.

$translator is a private property, provided at construction time. It is private, and only used in the processBadEmails() method. $translator may be turned into a parameter for processBadEmails(), and make the class slimmer.

```
class SendEmailToContact
{
    /**
     * @var TranslatorInterface
     */
    private $translator;

// Skipped code

    /**
     * SendEmailToContact constructor.
     *
     * @param MailHelper          $mailer
     * @param StatRepository      $statRepository
     * @param DoNotContact        $dncModel
```

(continues on next page)

(continued from previous page)

```
     * @param TranslatorInterface $translator
     */
    public function __construct(MailHelper $mailer, StatHelper $statHelper,␣
↪DoNotContact $dncModel, TranslatorInterface $translator)
    {
        $this->mailer     = $mailer;
        $this->statHelper = $statHelper;
        $this->dncModel   = $dncModel;
        $this->translator = $translator;
    }

// Skipped code

    /**
     * Add DNC entries for bad emails to get them out of the queue permanently.
     */
    protected function processBadEmails()
    {
        // Update bad emails as bounces
        if (count($this->badEmails)) {
            foreach ($this->badEmails as $contactId => $contactEmail) {
                $this->dncModel->addDncForContact(
                    $contactId,
                    ['email' => $this->emailEntityId],
                    DNC::BOUNCED,
                    $this->translator->trans('mautic.email.bounce.reason.bad_email'),
                    true,
                    false
                );
            }
        }
    }
```

Typo3

Property Could Be Local, in typo3/sysext/install/Classes/Updates/MigrateUrlTypesInPagesUpdate.php:28.

$urltypes is a private property, with a list of protocols for communicationss. It acts as a constant, being only read in the executeUpdate() method : constants may hold arrays. If this property has to evolve in the future, an accessor to update it will be necessary. Until then, this list may be hardcoded in the method.

```
/**
 * Merge URLs divided in pages.urltype and pages.url into pages.url
 * @internal This class is only meant to be used within EXT:install and is not part␣
↪of the TYPO3 Core API.
 */
class MigrateUrlTypesInPagesUpdate implements UpgradeWizardInterface
{
    private $urltypes = ['', 'http://', 'ftp://', 'mailto:', 'https://'];

// Skipped code

    /**
     * Moves data from pages.urltype to pages.url
```

(continues on next page)

(continued from previous page)

```
     *
     * @return bool
     */
    public function executeUpdate(): bool
    {
        foreach ($this->databaseTables as $databaseTable) {
            $connection = GeneralUtility::makeInstance(ConnectionPool::class)
                ->getConnectionForTable($databaseTable);

            // Process records that have entries in pages.urltype
            $queryBuilder = $connection->createQueryBuilder();
            $queryBuilder->getRestrictions()->removeAll();
            $statement = $queryBuilder->select('uid', 'urltype', 'url')
                ->from($databaseTable)
                ->where(
                    $queryBuilder->expr()->neq('urltype', 0),
                    $queryBuilder->expr()->neq('url', $queryBuilder->
↪createPositionalParameter(''))
                )
                ->execute();

            while ($row = $statement->fetch()) {
                $url = $this->urltypes[(int)$row['urltype']] . $row['url'];
                $updateQueryBuilder = $connection->createQueryBuilder();
                $updateQueryBuilder
                    ->update($databaseTable)
                    ->where(
                        $updateQueryBuilder->expr()->eq(
                            'uid',
                            $updateQueryBuilder->createNamedParameter($row['uid'],␣
↪\PDO::PARAM_INT)
                        )
                    )
                    ->set('url', $updateQueryBuilder->createNamedParameter($url),␣
↪false)
                    ->set('urltype', 0);
                $updateQueryBuilder->execute();
            }
        }
        return true;
    }
```

10.2.183 Too Many Native Calls

SPIP

Too Many Native Calls, in /ecrire/xml/analyser_dtd.php:58.

This expression counts 4 usages of count(), which is more than the default level of 3 PHP calls in one expression.

```
spip_log("Analyser DTD $avail $grammaire (" . spip_timer('dtd') . ") " . count($dtc->
↪macros) . ' macros, ' . count($dtc->elements) . ' elements, ' . count($dtc->
↪attributs) . " listes d'attributs, " . count($dtc->entites) . " entites")
```

10.2.184 Redefined Private Property

Zurmo

Redefined Private Property, in app/protected/modules/zurmo/models/OwnedCustomField.php:51.

The class OwnedCustomField is part of a large class tree : OwnedCustomField extends CustomField, CustomField extends BaseCustomField, BaseCustomField extends RedBeanModel, RedBeanModel extends BeanModel.

Since $canHaveBean is distinct in BeanModel and in OwnedCustomField, the public method getCanHaveBean() also had to be overloaded.

```
class OwnedCustomField extends CustomField
    {
        /**
         * OwnedCustomField does not need to have a bean because it stores no␣
↪attributes and has no relations
         * @see RedBeanModel::canHaveBean();
         * @var boolean
         */
        private static $canHaveBean = false;

/..../

        /**
         * @see RedBeanModel::getHasBean()
         */
        public static function getCanHaveBean()
        {
            if (get_called_class() == 'OwnedCustomField')
            {
                return self::$canHaveBean;
            }
            return parent::getCanHaveBean();
        }
```

10.2.185 Don’t Unset Properties

Vanilla

Don’t Unset Properties, in applications/dashboard/models/class.activitymodel.php:1073.

The _NotificationQueue property, in this class, is defined as an array. Here, it is destroyed, then recreated. The unset() is too much, as the assignation is sufficient to reset the array

```
/**
     * Clear notification queue.
     *
     * @since 2.0.17
     * @access public
     */
    public function clearNotificationQueue() {
        unset($this->_NotificationQueue);
        $this->_NotificationQueue = [];
    }
```

Typo3

Don't Unset Properties, in typo3/sysext/linkvalidator/Classes/Linktype/InternalLinktype.php:73.

The property errorParams is emptied by unsetting it. The property is actually defined in the above class, as an array. Until the next error is added to this list, any access to the error list has to be checked with isset(), or yield an 'Undefined' warning.

```
public function checkLink($url, $softRefEntry, $reference)
    {
        $anchor = '';
        $this->responseContent = true;
        // Might already contain values - empty it
        unset($this->errorParams);
//....

abstract class AbstractLinktype implements LinktypeInterface
{
    /**
     * Contains parameters needed for the rendering of the error message
     *
     * @var array
     */
    protected $errorParams = [];
```

10.2.186 Strtr Arguments

SuiteCrm

Strtr Arguments, in includes/vCard.php:221.

This code prepares incoming '$values' for extraction. The keys are cleaned then split with explode(). The '=' sign would stay, as strtr() can't remove it. This means that such keys won't be recognized later in the code, and gets omitted.

```
$values = explode(';', $value);
                    $key = strtoupper($keyvalue[0]);
                    $key = strtr($key, '=', '');
                    $key = strtr($key, ',', ';');
                    $keys = explode(';', $key);
```

10.2.187 Callback Needs Return

Contao

Callback Needs Return, in core-bundle/src/Resources/contao/modules/ModuleQuicklink.php:91.

The empty closure returns *null*. The array_flip() array has now all its values set to null, and reset, as intended. A better alternative is to use the array_fill_keys() function, which set a default value to every element of an array, once provided with the expected keys.

```
$arrPages = array_map(function () {}, array_flip($tmp));
```

Phpdocumentor

Callback Needs Return, in src/phpDocumentor/Plugin/ServiceProvider.php:24.

The array_walk() function is called on the plugin's list. Each element is registered with the application, but is not used directly : this is for later. The error mechanism is to throw an exception : this is the only expected feedback. As such, no return is expected. May be a 'foreach' loop would be more appropriate here, but this is syntactic sugar.

```
array_walk(
            $plugins,
            function ($plugin) use ($app) {
                /** @var Plugin $plugin */
                $provider = (strpos($plugin->getClassName(), '\') === false)
                    ? sprintf('phpDocumentor\Plugin\%s\ServiceProvider', $plugin->
↪getClassName())
                    : $plugin->getClassName();
                if (!class_exists($provider)) {
                    throw new \RuntimeException('Loading Service Provider for ' .
↪$provider . ' failed.');
                }

                try {
                    $app->register(new $provider($plugin));
                } catch (\InvalidArgumentException $e) {
                    throw new \RuntimeException($e->getMessage());
                }
            }
        );
```

10.2.188 Wrong Range Check

Dolibarr

Wrong Range Check, in htdocs/includes/phpoffice/PhpSpreadsheet/Spreadsheet.php:1484.

When $tabRatio is 1001, then the condition is valid, and the ratio accepted. The right part of the condition is not executed.

```
public function setTabRatio($tabRatio)
    {
        if ($tabRatio >= 0 || $tabRatio <= 1000) {
            $this->tabRatio = (int) $tabRatio;
        } else {
            throw new Exception('Tab ratio must be between 0 and 1000.');
        }
    }
```

WordPress

Wrong Range Check, in wp-includes/formatting.php:3634.

This condition may be easier to read as *$diff >= WEEK_IN_SECONDS && $diff < MONTH_IN_SECONDS*. When testing for outside this interval, using not is also more readable : *!($diff >= WEEK_IN_SECONDS && $diff < MONTH_IN_SECONDS)*.

```
} elseif ( $diff < MONTH_IN_SECONDS && $diff >= WEEK_IN_SECONDS ) {
            $weeks = round( $diff / WEEK_IN_SECONDS );
            if ( $weeks <= 1 ) {
                    $weeks = 1;
            }
            /* translators: Time difference between two dates, in weeks. %s: Number␣
↪of weeks */
            $since = sprintf( _n( '%s week', '%s weeks', $weeks ), $weeks );
```

10.2.189 Cant Instantiate Class

WordPress

Cant Instantiate Class, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

10.2.190 strpos() Too Much

WordPress

strpos() Too Much, in core/traits/Request/Server.php:127.

Instead of searching for `HTTP_`, it is faster to compare the first 5 chars to the literal `HTTP_`. In case of absence, this solution returns faster.

```
if (strpos($header, 'HTTP_') === 0) {
                            $header = substr($header, 5);
                    } elseif (strpos($header, 'CONTENT_') !== 0) {
                            continue;
                    }
```

10.2.191 Weak Typing

TeamPass

Weak Typing, in includes/libraries/Tree/NestedTree/NestedTree.php:100.

The is_null() test detects a special situation, that requires usage of default values. The 'else' handles every other situations, including when the $node is an object, or anything else. $this->getNode() will gain from having typehints : it may be NULL, or the results of mysqli_fetch_object() : a stdClass object. The expected properties of nleft and nright are not certain to be available.

```
public function getDescendants($id = 0, $includeSelf = false, $childrenOnly = false,
↪$unique_id_list = false)
    {
        global $link;
        $idField = $this->fields['id'];

        $node = $this->getNode($id);
```

(continues on next page)

(continued from previous page)

```
        if (is_null($node)) {
            $nleft = 0;
            $nright = 0;
            $parent_id = 0;
            $personal_folder = 0;
        } else {
            $nleft = $node->nleft;
            $nright = $node->nright;
            $parent_id = $node->$idField;
            $personal_folder = $node->personal_folder;
        }
```

10.2.192 Check JSON

Woocommerce

Check JSON, in includes/admin/helper/class-wc-helper-plugin-info.php:66.

In case the body is an empty string, this will be correctly decoded, but will yield an object with an empty-named property.

```
$results = json_decode( wp_remote_retrieve_body( $request ), true );
            if ( ! empty( $results ) ) {
                    $response = (object) $results;
            }

            return $response;
```

10.2.193 Bad Constants Names

PrestaShop

Bad Constants Names, in src/PrestaShopBundle/Install/Upgrade.php:214.

INSTALL_PATH is a valid name for a constant. __PS_BASE_URI__ is not a valid name.

```
require_once(INSTALL_PATH . 'install_version.php');
            // needed for upgrade before 1.5
            if (!defined('__PS_BASE_URI__')) {
                define('__PS_BASE_URI__', str_replace('//', '/', '/'.trim(preg_
↪replace('#/(install(-dev)?/upgrade)$#', '/', str_replace('\', '/', dirname($_SERVER[
↪'REQUEST_URI']))), '/').'/'));
            }
```

Zencart

Bad Constants Names, in zc_install/ajaxTestDBConnection.php:10.

A case where PHP needs help : if the PHP version is older than 5.3, then it is valid to compensate. Though, this __DIR__ has a fixed value, wherever it is used, while the official __DIR__ change from dir to dir.

```
if (!defined('__DIR__')) define('__DIR__', dirname(__FILE__));
```

10.2.194 Dont Mix ++

Contao

Dont Mix ++, in core-bundle/src/Resources/contao/drivers/DC_Table.php:1272.

Incrementing and multiplying at the same time.

```
$this->Database->prepare("UPDATE " . $this->strTable . " SET sorting=? WHERE id=?")
               ->execute(($count++ * 128), $objNewSorting->id);
```

Typo3

Dont Mix ++, in typo3/sysext/backend/Classes/Controller/SiteConfigurationController.php:74.

The post-increment is not readable at first glance.

```
foreach ($row['rootline'] as &$record) {
                $record['margin'] = $i++ * 20;
            }
```

10.2.195 Abstract Or Implements

Zurmo

Abstract Or Implements, in app/protected/extensions/zurmoinc/framework/views/MassEditProgressView.php:30.

The class MassEditProgressView extends ProgressView, which is an abstract class. That class defines one abstract method : abstract protected function headerLabelPrefixContent(). Yet, the class MassEditProgressView doesn't implements this method. This means that the class can't be instatiated, and indeed, it isn't. The class MassEditProgressView is subclassed, by the class MarketingListMembersMassSubscribeProgressView, which implements the method headerLabelPrefixContent(). As such, MassEditProgressView should be marked abstract, so as to prevent any instantiation attempt.

```
class MassEditProgressView extends ProgressView {
    /**/
}
```

10.2.196 Incompatible Signature Methods

SuiteCrm

Incompatible Signature Methods, in modules/Home/Dashlets/RSSDashlet/RSSDashlet.php:138.

The class in the RSSDashlet.php file has an 'array' typehint which is not in the parent Dashlet class. While both files compile separately, they yield a PHP warning when running : typehinting mismatch only yields a warning.

```
// File /modules/Home/Dashlets/RSSDashlet/RSSDashlet.php
    public function saveOptions(
        array $req
        )
    {

// File /include/Dashlets/Dashlets.php
    public function saveOptions( $req ) {
```

10.2.197 Ambiguous Visibilities

Typo3

Ambiguous Visibilities, in typo3/sysext/backend/Classes/Controller/NewRecordController.php:90.

$allowedNewTables is declared once protected and once public. $allowedNewTables is rare : 2 occurences. This may lead to confusion about access to this property.

```
class NewRecordController
{
/.. many lines../
    /**
     * @var array
     */
    protected $allowedNewTables;

class DatabaseRecordList
{
/..../
    /**
     * Used to indicate which tables (values in the array) that can have a
     * create-new-record link. If the array is empty, all tables are allowed.
     *
     * @var string[]
     */
    public $allowedNewTables = [];
```

10.2.198 Could Be Abstract Class

Edusoho

Could Be Abstract Class, in src/Biz/Task/Strategy/BaseStrategy.php:14.

BaseStrategy is extended by NormalStrategy, DefaultStrategy (Not shown here), but it is not instantiated itself.

```
class BaseStrategy {
    // Class code
}
```

shopware

Could Be Abstract Class, in engine/Shopware/Plugins/Default/Core/PaymentMethods/Components/GenericPaymentMethod.php:31.

A ‘Generic’ class sounds like a class that could be ‘abstract’.

```
class GenericPaymentMethod extends BasePaymentMethod {
    // More class code
}
```

10.2.199 Continue Is For Loop

XOOPS

Continue Is For Loop, in htdocs/kernel/object.php:711.

break is used here for cases, unless the case includes a if/then structures, in which it becomes a continue. It really should be a break.

```
foreach ($this->vars as $k => $v) {
            $cleanv = $v['value'];
            if (!$v['changed']) {
            } else {
                $cleanv = is_string($cleanv) ? trim($cleanv) : $cleanv;
                switch ($v['data_type']) {
                    case XOBJ_DTYPE_TIMESTAMP:
                        $cleanv = !is_string($cleanv) && is_numeric($cleanv) ? date(_
↪DBTIMESTAMPSTRING, $cleanv) : date(_DBTIMESTAMPSTRING, strtotime($cleanv));
                        break;
                    case XOBJ_DTYPE_TIME:
                        $cleanv = !is_string($cleanv) && is_numeric($cleanv) ? date(_
↪DBTIMESTRING, $cleanv) : date(_DBTIMESTRING, strtotime($cleanv));
                        break;
                    case XOBJ_DTYPE_DATE:
                        $cleanv = !is_string($cleanv) && is_numeric($cleanv) ? date(_
↪DBDATESTRING, $cleanv) : date(_DBDATESTRING, strtotime($cleanv));
                        break;
                    case XOBJ_DTYPE_TXTBOX:
                        if ($v['required'] && $cleanv != '0' && $cleanv == '') {
                            $this->setErrors(sprintf(_XOBJ_ERR_REQUIRED, $k));
                            continue 2;
                        }
                        if (isset($v['maxlength']) && strlen($cleanv) > (int)$v[
↪'maxlength']) {
                            $this->setErrors(sprintf(_XOBJ_ERR_SHORTERTHAN, $k, (int)
↪$v['maxlength']));
                            continue 2;
                        }
```

10.2.200 Wrong Access Style to Property

HuMo-Gen

Wrong Access Style to Property, in wp-admin/includes/misc.php:74.

lame_binary_path is a static property, but it is accessed as a normal property in the exception call, while it is checked with a valid syntax.

```
protected function wavToMp3($data)
    {
        if (!file_exists(self::$lame_binary_path) || !is_executable(self::$lame_
↪binary_path)) {
            throw new Exception('Lame binary  . $this->lame_binary_path .  does not␣
↪exist or is not executable');
        }
```

10.2.201 Method Could Be Static

FuelCMS

Method Could Be Static, in fuel/modules/fuel/models/Fuel_assets_model.php:240.

This method makes no usage of $this : it only works on the incoming argument, $file. This may even be a function.

```
public function get_file($file)
    {
            // if no extension is provided, then we determine that it needs to be␣
↪decoded
            if (strpos($file, '.') === FALSE)
            {
                    $file = uri_safe_decode($file);
            }
            return $file;
    }
```

ExpressionEngine

Method Could Be Static, in system/ee/legacy/libraries/Upload.ph:859.

This method returns the list of mime type, by using a hidden global value : ee() is a functioncall that give access to the external storage of values.

```
/**
     * List of Mime Types
     *
     * This is a list of mime types.  We use it to validate
     * the allowed types set by the developer
     *
     * @param       string
     * @return      string
     */
    public function mimes_types($mime)
    {
            ee()->load->library('mime_type');
            return ee()->mime_type->isSafeForUpload($mime);
    }
```

10.2.202 Possible Missing Subpattern

phpMyAdmin

Possible Missing Subpattern, in libraries/classes/Advisor.php:557.

The last capturing subpattern is `( \[(.*)\])?` and it is optional. Indeed, when the pattern succeed, the captured values are stored in `$match`. Yet, the code checks for the existence of `$match[3]` before using it.

```
if (preg_match("/rule\s'(.*)'( \[(.*)\])?$/", $line, $match)) {
                    $ruleLine = 1;
                    $ruleNo++;
                    $rules[$ruleNo] = ['name' => $match[1]];
                    $lines[$ruleNo] = ['name' => $i + 1];
                    if (isset($match[3])) {
                        $rules[$ruleNo]['precondition'] = $match[3];
                        $lines[$ruleNo]['precondition'] = $i + 1;
                    }
```

SPIP

Possible Missing Subpattern, in ecrire/inc/filtres_dates.php:73.

This code avoid the PHP notice by padding the resulting array (see comment in French : eviter === avoid)

```
if (preg_match("#^([12][0-9]{3}[-/][01]?[0-9])([-/]00)?( [-0-9:]+)?$#", $date,
↪$regs)) {
                            $regs = array_pad($regs, 4, null); // eviter notice php
                            $date = preg_replace("@/@", "-", $regs[1]) . "-00" .
↪$regs[3];
                    } else {
                            $date = date("Y-m-d H:i:s", strtotime($date));
                    }
```

10.2.203 Overwritten Source And Value

ChurchCRM

Overwritten Source And Value, in edusoho/vendor/symfony/symfony/src/Symfony/Component/VarDumper/Dumper/CliDumper.php:194

$str is actually processed as an array (string of characters), and it is also modified along the way.

```
foreach ($str as $str) {
                if ($i < $m) {
                    $str .= \n;
                }
                if (0 < $this->maxStringWidth && $this->maxStringWidth < $len = mb_
↪strlen($str, 'UTF-8')) {
                    $str = mb_substr($str, 0, $this->maxStringWidth, 'UTF-8');
                    $lineCut = $len - $this->maxStringWidth;
                }
                //.... More code
```

ExpressionEngine

Overwritten Source And Value, in system/ee/EllisLab/ExpressionEngine/Service/Theme/ThemeInstaller.php:595.

Looping over $filename.

```
foreach (directory_map($to_dir) as $directory => $filename)
                    {
                            if (is_string($directory))
                            {
                                    foreach ($filename as $filename)
                                    {
                                            unlink($to_dir.$directory.'/'.$filename);
                                    }

                                    @rmdir($to_dir.$directory);
                            }
                            else
                            {
                                    unlink($to_dir.$filename);
                            }
                    }
```

10.2.204 Incompatible Signature Methods With Covariance

SuiteCrm

Incompatible Signature Methods With Covariance, in modules/Home/Dashlets/RSSDashlet/RSSDashlet.php:138.

The class in the RSSDashlet.php file has an 'array' typehint which is not in the parent Dashlet class. While both files compile separately, they yield a PHP warning when running : typehinting mismatch only yields a warning.

```
// File /modules/Home/Dashlets/RSSDashlet/RSSDashlet.php
    public function saveOptions(
        array $req
        )
    {

// File /include/Dashlets/Dashlets.php
    public function saveOptions( $req ) {
```

10.2.205 Could Be Private Class Constant

Phinx

Could Be Private Class Constant, in src/Phinx/Db/Adapter/MysqlAdapter.php:46.

The code includes a fair number of class constants. The one listed here are only used to define TEXT columns in MySQL, with their maximal size. Since they are only intented to be used by the MySQL driver, they may be private.

```
class MysqlAdapter extends PdoAdapter implements AdapterInterface
{

//.....
```

(continues on next page)

(continued from previous page)

```
    const TEXT_SMALL   = 255;
    const TEXT_REGULAR = 65535;
    const TEXT_MEDIUM  = 16777215;
    const TEXT_LONG    = 4294967295;
```

10.2.206 Disconnected Classes

WordPress

Disconnected Classes, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

10.2.207 Wrong Class Name Case

WordPress

Wrong Class Name Case, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

10.2.208 One Letter Functions

ThinkPHP

One Letter Functions, in ThinkPHP/Mode/Api/functions.php:859.

There are also the functions C, E, G… The applications is written in a foreign language, which may be a base for non-significant function names.

```
function F($name, $value = '', $path = DATA_PATH)
```

Cleverstyle

One Letter Functions, in core/drivers/DB/PostgreSQL.php:71.

There is also function f(). Those are actually overwritten methods. From the documentation, q() is for query, and f() is for fetch. Those are short names for frequently used functions.

```
public function q ($query, ...$params) {
```

10.2.209 __debugInfo() Usage

Dolibarr

__debugInfo() Usage, in htdocs/includes/stripe/lib/StripeObject.php:108.

_values is a private property from the Stripe Class. The class contains other objects, but only _values are displayed with var_dump.

```
// Magic method for var_dump output. Only works with PHP >= 5.6
    public function __debugInfo()
    {
        return $this->_values;
    }
```

10.2.210 PHP7 Dirname

OpenConf

PHP7 Dirname, in include.php:61.

Since PHP 7.0, dirname(, 2); does the job.

```
$OC_basepath = dirname(dirname($_SERVER['PHP_SELF']));
```

MediaWiki

PHP7 Dirname, in includes/installer/Installer.php:1173.

Since PHP 7.0, dirname(, 2); does the job.

```
protected function envPrepPath() {
            global $IP;
            $IP = dirname( dirname( __DIR__ ) );
            $this->setVar( 'IP', $IP );
    }
```

10.2.211 Avoid set_error_handler $context Argument

shopware

Avoid set_error_handler $context Argument, in engine/Shopware/Plugins/Default/Core/ErrorHandler/Bootstrap.php:162.

The registered handler is a local method, called `errorHandler`, which has 6 arguments, and relays those 6 arguments to set_error_handler().

```
public function registerErrorHandler($errorLevel = E_ALL)
    {
        // Only register once.  Avoids loop issues if it gets registered twice.
        if (self::$_registeredErrorHandler) {
            set_error_handler([$this, 'errorHandler'], $errorLevel);
```

(continues on next page)

(continued from previous page)

```
            return $this;
        }

        self::$_origErrorHandler = set_error_handler([$this, 'errorHandler'],
↪$errorLevel);
        self::$_registeredErrorHandler = true;

        return $this;
    }
```

Vanilla

Avoid set_error_handler $context Argument, in library/core/functions.error.php:747.

Gdn_ErrorHandler is a function that requires 6 arguments.

```
set_error_handler('Gdn_ErrorHandler', E_ALL & ~E_STRICT)
```

10.2.212 Unused Private Properties

OpenEMR

Unused Private Properties, in entities/User.php:46.

This class has a long list of private properties. It also has an equally long (minus one) list of accessors, and a __toString() method which exposes all of them. $oNotes is the only one never mentionned anywhere.

```
class User
{
    /**
     * @Column(name=id, type=integer)
     * @GeneratedValue(strategy=AUTO)
     */
    private $id;

    /**
     * @OneToMany(targetEntity=ONote, mappedBy=user)
     */
    private $oNotes;
```

---

phpadsnew

Unused Private Properties, in lib/OA/Admin/UI/component/Form.php:23.

$dispatcher is never used anywhere.

```
class OA_Admin_UI_Component_Form
    extends HTML_QuickForm
{
    private $dispatcher;
```

10.2.213 Unused Functions

Woocommerce

Unused Functions, in includes/wc-core-functions.php:2124.

wc_is_external_resource() is unused. This is not obvious immediately, since there is a call from wc_get_relative_url(). Yet since wc_get_relative_url() itself is never used, then it is a dead function. As such, since wc_is_external_resource() is only called by this first function, it also dies, even though it is called in the code.

```
/**
 * Make a URL relative, if possible.
 *
 * @since 3.2.0
 * @param string $url URL to make relative.
 * @return string
 */
function wc_get_relative_url( $url ) {
    return wc_is_external_resource( $url ) ? $url : str_replace( array( 'http://',
↪'https://' ), '//', $url );
}

/**
 * See if a resource is remote.
 *
 * @since 3.2.0
 * @param string $url URL to check.
 * @return bool
 */
function wc_is_external_resource( $url ) {
    $wp_base = str_replace( array( 'http://', 'https://' ), '//', get_home_url( null,
↪'/', 'http' ) );

    return strstr( $url, '://' ) && ! strstr( $url, $wp_base );
}
```

Piwigo

Unused Functions, in admin/include/functions.php:2167.

get_user_access_level_html_options() is unused and can’t be find in the code.

```
/**
 * Returns access levels as array used on template with html_options functions.
 *
 * @param int $MinLevelAccess
 * @param int $MaxLevelAccess
```

(continues on next page)

(continued from previous page)

```
 * @return array
 */
function get_user_access_level_html_options($MinLevelAccess = ACCESS_FREE,
↪$MaxLevelAccess = ACCESS_CLOSED)
{
  $tpl_options = array();
  for ($level = $MinLevelAccess; $level <= $MaxLevelAccess; $level++)
  {
    $tpl_options[$level] = l10n(sprintf('ACCESS_%d', $level));
  }
  return $tpl_options;
}
```

10.2.214 Unused Interfaces

Tine20

Unused Interfaces, in tine20/Tinebase/User/LdapPlugin/Interface.php:20.

Tinebase_User_LdapPlugin_Interface is mentioned as a type for a property, in a php doc document. Typehinted properties are available since PHP 7.4

```
interface Tinebase_User_LdapPlugin_Interface {

//----------
// in tine20/Tinebase/User/ActiveDirectory.php
/** @var Tinebase_User_LdapPlugin_Interface $plugin */
```

10.2.215 Exception Order

Woocommerce

Exception Order, in includes/api/v1/class-wc-rest-products-controller.php:787.

This try/catch expression is able to catch both WC_Data_Exception and WC_REST_Exception.

In another file, /includes/api/class-wc-rest-exception.php, we find that WC_REST_Exception extends WC_Data_Exception (class WC_REST_Exception extends WC_Data_Exception {}). So WC_Data_Exception is more general, and a WC_REST_Exception exception is caught with WC_Data_Exception Exception. The second catch should be put in first.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
try {
                    $product_id = $this->save_product( $request );
                    $post       = get_post( $product_id );
                    $this->update_additional_fields_for_object( $post, $request );
                    $this->update_post_meta_fields( $post, $request );

                    /**
                     * Fires after a single item is created or updated via the REST␣
↪API.
                     *
```

(continues on next page)

(continued from previous page)

```
                         * @param WP_Post          $post      Post data.
                         * @param WP_REST_Request $request   Request object.
                         * @param boolean         $creating  True when creating item,␣
↪false when updating.
                         */
                        do_action( 'woocommerce_rest_insert_product', $post, $request,␣
↪false );
                        $request->set_param( 'context', 'edit' );
                        $response = $this->prepare_item_for_response( $post, $request );

                        return rest_ensure_response( $response );
                } catch ( WC_Data_Exception $e ) {
                        return new WP_Error( $e->getErrorCode(), $e->getMessage(), $e->
↪getErrorData() );
                } catch ( WC_REST_Exception $e ) {
                        return new WP_Error( $e->getErrorCode(), $e->getMessage(), array(
↪'status' => $e->getCode() ) );
                }
```

10.2.216 Rethrown Exceptions

PrestaShop

Rethrown Exceptions, in classes/webservice/WebserviceOutputBuilder.php:731.

The setSpecificField method catches a WebserviceException, representing an issue with the call to the webservice. However, that piece of information is lost, and the exception is rethrown immediately, without any action.

```
public function setSpecificField($object, $method, $field_name, $entity_name)
    {
        try {
                $this->validateObjectAndMethod($object, $method);
        } catch (WebserviceException $e) {
                throw $e;
        }

        $this->specificFields[$field_name] = array('entity'=>$entity_name, 'object
↪' => $object, 'method' => $method, 'type' => gettype($object));
        return $this;
    }
```

10.2.217 Slow Functions

ChurchCRM

Slow Functions, in src/Reports/PrintDeposit.php:35.

You may replace this with a isset() : $_POST can't contain a NULL value, unless it was set by the script itself.

```
array_key_exists("report_type", $_POST);
```

SuiteCrm

Slow Functions, in include/json_config.php:242.

This is a equivalent for nl2br()

```
preg_replace("/\r\n/", "<BR>", $focus->$field)
```

10.2.218 Joining file()

WordPress

Joining file(), in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

SPIP

Joining file(), in ecrire/inc/install.php:109.

When the file is not accessible, file() returns null, and can’t be processed by join().

```
$s = @join('', file($file));
```

ExpressionEngine

Joining file(), in ExpressionEngine_Core2.9.2/system/expressionengine/libraries/simplepie/idn/idna_convert.class.php:100.

join(‘’,) is used as a replacement for file_get_contents(), which was introduced in PHP 4.3.0.

```
if (function_exists('file_get_contents')) {
    $this->NP = unserialize(file_get_contents(dirname(__FILE__).'/npdata.ser'));
} else {
    $this->NP = unserialize(join('', file(dirname(__FILE__).'/npdata.ser')));
}
```

PrestaShop

Joining file(), in classes/module/Module.php:2972.

implode(‘’,) is probably not the slowest part in these lines.

```
$override_file = file($override_path);

eval(preg_replace(array('#^\s*<\?(?:php)?#', '#class\s+'.$classname.'\s+extends\s+([a-
↪z0-9_]+)(\s+implements\s+([a-z0-9_]+))?#i'), array(' ', 'class '.$classname.
↪'OverrideOriginal_remove'.$uniq), implode('', $override_file)));
$override_class = new ReflectionClass($classname.'OverrideOriginal_remove'.$uniq);

$module_file = file($this->getLocalPath().'override/'.$path);
eval(preg_replace(array('#^\s*<\?(?:php)?#', '#class\s+'.$classname.
↪'(\s+extends\s+([a-z0-9_]+)(\s+implements\s+([a-z0-9_]+))?)?#i'), array(' ', 'class
↪'.$classname.'Override_remove'.$uniq), implode('', $module_file)));
```

10.2.219 Simplify Regex

Zurmo

Simplify Regex, in app/protected/core/components/Browser.php:73.

Here, strpos() or stripos() is a valid replacement.

```
preg_match('/opera/', $userAgent)
```

OpenConf

Simplify Regex, in openconf/include.php:964.

%e is not a special char for PCRE regex, although it look like it. It is a special char for date() or printf(). This preg_replace() may be upgraded to str_replace()

```
$conv = iconv($cp, 'utf-8', strftime(preg_replace("/\%e/", '%#d', $format), $time));
```

10.2.220 Make One Call With Array

HuMo-Gen

Make One Call With Array, in admin/include/kcfinder/lib/helper_text.php:47.

The three calls to str_replace() could be replaced by one, using array arguments. Nesting the calls doesn't reduce the number of calls.

```
static function jsValue($string) {
        return
            preg_replace('/\r?\n/', "\n",
            str_replace('"', "\\"",
            str_replace("'", "\'",
            str_replace("\", "\\",
        $string))));
    }
```

Edusoho

Make One Call With Array, in src/AppBundle/Common/StringToolkit.php:55.

Since str_replace is already using an array, the second argument must also be an array, with repeated empty strings. That syntax allows adding the ‘&nbsp;’ and ‘ ‘ to those arrays. Note also that trim() should be be called early, but since some of the replacing may generate terminal spaces, it should be kept as is.

```
$text = strip_tags($text);

        $text = str_replace(array(\n, \r, \t), '', $text);
        $text = str_replace('&nbsp;', ' ', $text);
        $text = trim($text);
```

10.2.221 No Count With 0

Contao

No Count With 0, in system/modules/repository/classes/RepositoryManager.php:1148.

If $elist contains at least one element, then it is not empty().

```
$ext->found = count($elist)>0;
```

WordPress

No Count With 0, in wp-admin/includes/misc.php:74.

$build or $signature are empty at that point, no need to calculate their respective length.

```
// Check for zero length, although unlikely here
    if (strlen($built) == 0 || strlen($signature) == 0) {
      return false;
    }
```

10.2.222 time() Vs strtotime()

Woocommerce

time() Vs strtotime(), in includes/class-wc-webhook.php:384.

time() would be faster here, as an entropy generator. Yet, it would still be better to use an actual secure entropy generator, like random_byte or random_int. In case of older version, microtime() would yield better entropy.

```
public function get_new_delivery_id() {
            // Since we no longer use comments to store delivery logs, we generate a␣
↪unique hash instead based on current time and webhook ID.
            return wp_hash( $this->get_id() . strtotime( 'now' ) );
    }
```

10.2.223 Getting Last Element

Thelia

Getting Last Element, in /core/lib/Thelia/Core/Security/AccessManager.php:61.

This code extract the last element with array_slice (position -1) as an array, then get the element in the array with current().

```
current(\array_slice(self::$accessPows, -1, 1, true))
```

10.2.224 Avoid glob() Usage

Phinx

Avoid glob() Usage, in src/Phinx/Migration/Manager.php:362.

glob() searches for a list of files in the migration folder. Those files are not known, but they have a format, as checked later with the regex : a combinaison of `FilesystemIterator` and `RegexIterator` would do the trick too.

```
$phpFiles = glob($config->getMigrationPath() . DIRECTORY_SEPARATOR . '*.php');

            // filter the files to only get the ones that match our naming scheme
            $fileNames = array();
            /** @var AbstractMigration[] $versions */
            $versions = array();

            foreach ($phpFiles as $filePath) {
                if (preg_match('/([0-9]+)_([_a-z0-9]*).php/', basename($filePath))) {
```

NextCloud

Avoid glob() Usage, in lib/private/legacy/helper.php:185.

Recursive copy of folders, based on scandir(). `DirectoryIterator` and `FilesystemIterator` would do the same without the recursion.

```
static function copyr($src, $dest) {
            if (is_dir($src)) {
                    if (!is_dir($dest)) {
                            mkdir($dest);
                    }
                    $files = scandir($src);
                    foreach ($files as $file) {
                            if ($file != "." && $file != "..") {
                                    self::copyr("$src/$file", "$dest/$file");
                            }
                    }
            } elseif (file_exists($src) && !\OC\Files\Filesystem::isFileBlacklisted(
↪$src)) {
                    copy($src, $dest);
            }
    }
```

10.2.225 Avoid Concat In Loop

SuiteCrm

Avoid Concat In Loop, in include/export_utils.php:433.

$line is build in several steps, then then final version is added to $content. It would be much faster to make $content an array, and implode it once after the loop.

```
foreach($records as $record)
        {
            $line = implode("\"" . getDelimiter() . "\"", $record);
            $line = "\"" . $line;
            $line .= "\"\r\n";
            $line = parseRelateFields($line, $record, $customRelateFields);
            $content .= $line;
        }
```

ThinkPHP

Avoid Concat In Loop, in ThinkPHP/Common/functions.php:720.

The foreach loop appends the $name and builds a fully qualified name.

```
if (!C('APP_USE_NAMESPACE')) {
        $class = parse_name($name, 1);
        import($module . '/' . $layer . '/' . $class . $layer);
    } else {
        $class = $module . '\' . $layer;
        foreach ($array as $name) {
            $class .= '\' . parse_name($name, 1);
        }
        //
        if ($extend) {
            //
            $class = $extend . '\' . $class;
        }
    }
    return $class . $layer;
```

10.2.226 Use pathinfo() Arguments

Zend-Config

Use pathinfo() Arguments, in src/Factory.php:74:90.

The *$filepath* is broken into pieces, and then, only the ‘extension’ part is used. With the PATHINFO_EXTENSION constant used as a second argument, only this value could be returned.

```
$pathinfo = pathinfo($filepath);

        if (! isset($pathinfo['extension'])) {
            throw new Exception\RuntimeException(sprintf(
```

(continues on next page)

(continued from previous page)

```
                'Filename "%s" is missing an extension and cannot be auto-detected',
                $filename
            ));
        }

        $extension = strtolower($pathinfo['extension']);
        // Only $extension is used beyond that point
```

ThinkPHP

Use pathinfo() Arguments, in ThinkPHP/Extend/Library/ORG/Net/UploadFile.class.php:508.

Without any other check, pathinfo() could be used with PATHINFO_EXTENSION.

```
private function getExt($filename) {
        $pathinfo = pathinfo($filename);
        return $pathinfo['extension'];
    }
```

10.2.227 Substring First

SPIP

Substring First, in ecrire/inc/filtres.php:1694.

The code first makes everything uppercase, including the leading and trailing spaces, and then, removes them : it would be best to swap those operations. Note that spip_substr() is not considered in this analysis, but with SPIP knowledge, it could be moved inside the calls.

```
function filtre_initiale($nom) {
    return spip_substr(trim(strtoupper(extraire_multi($nom))), 0, 1);
}
```

PrestaShop

Substring First, in admin-dev/filemanager/include/utils.php:197.

dirname() reduces the string (or at least, keeps it the same size), so it more efficient to have it first.

```
dirname(str_replace(' ', '~', $str))
```

10.2.228 Slice Arrays First

WordPress

Slice Arrays First, in modules/InboundEmail/InboundEmail.php:1080.

Instead of reading ALL the keys, and then, keeping only the first fifty, why not read the 50 first items from the array, and then extract the keys?

```
$results = array_slice(array_keys($diff), 0 ,50);
```

10.2.229 Double array_flip()

NextCloud

Double array_flip(), in lib/public/AppFramework/Http/EmptyContentSecurityPolicy.php:372.

The array $allowedScriptDomains is flipped, to unset 'self', then, unflipped (or flipped again), to restore its initial state. Using array_keys() or array_search() would yield the needed keys for unsetting, at a lower cost.

```
if(is_string($this->useJsNonce)) {
                            $policy .= '\'nonce-'.base64_encode($this->useJsNonce).'\'
↪';
                            $allowedScriptDomains = array_flip($this->
↪allowedScriptDomains);
                            unset($allowedScriptDomains['\'self\'']);
                            $this->allowedScriptDomains = array_flip(
↪$allowedScriptDomains);
                            if(count($allowedScriptDomains) !== 0) {
                                    $policy .= ' ';
                            }
                    }
```

10.2.230 Closure Could Be A Callback

Tine20

Closure Could Be A Callback, in tine20/Tinebase/Convert/Json.php:318.

is_scalar() is sufficient here.

```
$value = array_filter($value, function ($val) { return is_scalar($val); });
```

NextCloud

Closure Could Be A Callback, in apps/files_sharing/lib/ShareBackend/Folder.php:114.

$qb is the object for the methodcall, passed via use. The closure may have been replaced with array($qb, 'createNamedParameter').

```
$parents = array_map(function($parent) use ($qb) {
                            return $qb->createNamedParameter($parent);
                    }, $parents);
```

10.2.231 Isset() On The Whole Array

Tine20

Isset() On The Whole Array, in tine20/Crm/Model/Lead.php:208.

Only the second call is necessary : it also includes the first one.

```
isset($relation['related_record']) && isset($relation['related_record']['n_fileas'])
```

ExpressionEngine

Isset() On The Whole Array, in system/ee/legacy/libraries/Form_validation.php:1487.

This is equivalent to *isset($this->_field_data[$field], $this->_field_data[$field]['postdata'])*, and the second call may be skipped.

```
!isset($this->_field_data[$field]) OR !isset($this->_field_data[$field]['postdata'])
```

10.2.232 Compare Hash

Traq

Compare Hash, in src/Models/User.php:105.

This code should also avoid using SHA1.

```
sha1($password) == $this->password
```

LiveZilla

Compare Hash, in livezilla/_lib/objects.global.users.inc.php:1391.

This code is using the stronger SHA256 but compares it to another string. $_token may be non-empty, and still be comparable to 0.

```
function IsValidToken($_token)
{
    if(!empty($_token))
        if(hash("sha256",$this->Token) == $_token)
            return true;
    return false;
}
```

10.2.233 Register Globals

TeamPass

Register Globals, in api/index.php:25.

The API starts with security features, such as the whitelist(). The whitelist applies to IP addresses, so the query string is not sanitized. Then, the QUERY_STRING is parsed, and creates a lot of new global variables.

```
teampass_whitelist();

parse_str($_SERVER['QUERY_STRING']);
$method = $_SERVER['REQUEST_METHOD'];
$request = explode("/", substr(@$_SERVER['PATH_INFO'], 1));
```

XOOPS

Register Globals, in htdocs/modules/system/admin/images/main.php:33:33.

This code only exports the POST variables as globals. And it does clean incoming variables, but not all of them.

```
// Check users rights
if (!is_object($xoopsUser) || !is_object($xoopsModule) || !$xoopsUser->isAdmin(
↪$xoopsModule->mid())) {
    exit(_NOPERM);
}

//  Check is active
if (!xoops_getModuleOption('active_images', 'system')) {
    redirect_header('admin.php', 2, _AM_SYSTEM_NOTACTIVE);
}

if (isset($_POST)) {
    foreach ($_POST as $k => $v) {
        ${$k} = $v;
    }
}

// Get Action type
$op = system_CleanVars($_REQUEST, 'op', 'list', 'string');
```

10.2.234 Safe Curl Options

OpenConf

Safe Curl Options, in openconf/include.php:703.

The function that holds that code is only used to call openconf.com, over http, while openconf.com is hosted on https, nowadays. This may be a sign of hard to access certificates.

```
$ch = curl_init();
                    curl_setopt($ch, CURLOPT_URL, $f);
                    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
                    curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
                    curl_setopt($ch, CURLOPT_AUTOREFERER, true);
                    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
                    curl_setopt($ch, CURLOPT_MAXREDIRS, 5);
                    curl_setopt($ch, CURLOPT_HEADER, false);
                    $s = curl_exec($ch);
                    curl_close($ch);
                    return($s);
```

10.2.235 Unserialize Second Arg

Piwigo

Unserialize Second Arg, in admin/configuration.php:491.

unserialize() extracts information from the $conf variable : this variable is read from a configuration file. It is later tested to be an array, whose index may not be all set (@$disabled[$type];). It would be safer to make $disabled an object, add the class to unserialize, and set default values to the needed properties/index.

```
$disabled = @unserialize(@$conf['disabled_derivatives']);
```

LiveZilla

Unserialize Second Arg, in livezilla/_lib/objects.global.inc.php:2600.

unserialize() only extract a non-empty value here. But its content is not checked. It is later used as an array, with multiple index.

```
$this->Customs = (!empty($_row["customs"])) ? @unserialize($_row["customs"]) :␣
↪array();
```

10.2.236 Encoded Simple Letters

Zurmo

Encoded Simple Letters, in yii/framework/web/CClientScript.php:783.

This actually decodes into a copyright notice.

‘function cleanAndSanitizeScriptHeader(& $output)

> { $requiredOne = <span>Copyright &#169; Zurmo Inc., 2013. All rights reserved.;. . . ’

```
eval(\x66\x75\x6e\x63\x74\x69\x6f\x6e\x20\x63\x6c\x65\x61\x6e\x41\x6e\x64\x53\x61\x6e\x69\x74\x69\x7a
↪.
   ␣
↪\x69\x70\x74\x48\x65\x61\x64\x65\x72\x28\x26\x20\x24\x6f\x75\x74\x70\x75\x74\x29\x0d\x0a\x20\x20\x2
↪.
   ␣
↪\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x7b\x0d\x0a\x20\x20\x20\x2
↪.
   ␣
↪\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x24\x72\x65\x7
↪.
     // several more lines like that
```

10.2.237 Mkdir Default

Mautic

Mkdir Default, in app/bundles/CoreBundle/Helper/AssetGenerationHelper.php:120.

This code is creating some directories for Javascript or CSS (from the directories names) : those require universal reading access, but probably no execution nor writing access. 0711 would be sufficient in this case.

```
//combine the files into their corresponding name and put in the root media folder
                if ($env == 'prod') {
                    $checkPaths = [
                        $assetsFullPath,
                        $assetsFullPath/css,
                        $assetsFullPath/js,
                    ];
                    array_walk($checkPaths, function ($path) {
                        if (!file_exists($path)) {
                            mkdir($path);
                        }
                    });
```

OpenEMR

Mkdir Default, in interface/main/backuplog.php:27.

If $BACKUP_EVENTLOG_DIR is a backup for an event log, this should be stored out of the web server reach, with low rights, beside the current user. This is part of a CLI PHP script.

```
mkdir($BACKUP_EVENTLOG_DIR)
```

10.2.238 Phpinfo

Dolphin

Phpinfo, in Dolphin-v.7.3.5/install/exec.php:4.

An actual phpinfo(), available during installation. Note that the phpinfo() is actually triggered by a hidden POST variable.

```
<?php

    if (!empty($_POST['phpinfo']))
        phpinfo();
    elseif (!empty($_POST['gdinfo']))
        echo '<pre>' . print_r(gd_info(), true) . '</pre>';

?>
<center>

    <form method=post>
        <input type=submit name=phpinfo value="PHP Info">
    </form>
    <form method=post>
        <input type=submit name=gdinfo value="GD Info">
    </form>

</center>
```

10.2.239 Configure Extract

Zurmo

Configure Extract, in app/protected/modules/marketing/utils/GlobalMarketingFooterUtil.php:127.

This code intent to overwrite *$hash* and *$preview* : it is even literally in the code. The overwrite is intended too, and could even skip the initialisation of the variables. Although the compact()/extract() combinaison is safe as now, it could be safer to only relay the array index, instead of extracting the variables here.

```
public static function resolveManageSubscriptionsUrlByArray(array $queryStringArray,
↪$preview = false)
        {
            $hash = $preview = null;
            extract(static::resolvePreviewAndHashFromArray($queryStringArray));
            return static::resolveManageSubscriptionsUrl($hash, $preview);
        }

// Also with :
        protected static function resolvePreviewAndHashFromArray(array
↪$queryStringArray)
        {
            $preview    = static::resolvePreviewFromArray($queryStringArray);
            $hash       = static::resolveHashByArray($queryStringArray);
            return compact('hash', 'preview');
        }
```

Dolibarr

Configure Extract, in htdocs/includes/restler/framework/Luracast/Restler/Format/HtmlFormat.php:224.

The extract() has been cleverly set in a closure, with a limited scope. The potential overwrite may impact existing variables, such as *$_*, *$nav*, *$form*, and *$data* itself. This may impact the following including. Using EXTR_SKIP would give existing variables priority, and avoid interference.

```
$template = function ($view) use ($data, $path) {
            $form = function () {
                return call_user_func_array(
                    'Luracast\Restler\UI\Forms::get',
                    func_get_args()
                );
            };
            if (!isset($data['form']))
                $data['form'] = $form;
            $nav = function () {
                return call_user_func_array(
                    'Luracast\Restler\UI\Nav::get',
                    func_get_args()
                );
            };
            if (!isset($data['nav']))
                $data['nav'] = $nav;

            $_ = function () use ($data, $path) {
```

(continues on next page)

(continued from previous page)

```
                extract($data);
                $args = func_get_args();
                $task = array_shift($args);
                switch ($task) {
                    case 'require':
                    case 'include':
                        $file = $path . $args[0];
                        if (is_readable($file)) {
                            if (
                                isset($args[1]) &&
                                ($arrays = Util::nestedValue($data, $args[1]))
                            ) {
                                $str = '';
                                foreach ($arrays as $arr) {
                                    extract($arr);
                                    $str .= include $file;
                                }
                                return $str;
                            } else {
                                return include $file;
                            }
                        }
                        break;
                    case 'if':
                        if (count($args) < 2)
                            $args[1] = '';
                        if (count($args) < 3)
                            $args[2] = '';
                        return $args[0] ? $args[1] : $args[2];
                        break;
                    default:
                        if (isset($data[$task]) && is_callable($data[$task]))
                            return call_user_func_array($data[$task], $args);
                }
                return '';
            };
            extract($data);
            return @include $view;
        };
```

10.2.240 Property Variable Confusion

PhpIPAM

Property Variable Confusion, in functions/classes/class.Admin.php:16.

There is a property called '$users'. It is easy to mistake $this->users and $users. Also, it seems that $this->users may be used as a cache system, yet it is not employed here.

```
/**
     * (array of objects) to store users, user id is array index
     *
     * @var mixed
     * @access public
     */
```

(continues on next page)

(continued from previous page)

```
    public $users;

////////////

    /**
     * Fetches all users that are in group
     *
     * @access public
     * @return array of user ids
     */
    public function group_fetch_users ($group_id) {
            $out = array ();
            # get all users
            $users = $this->fetch_all_objects(users);
            # check if $gid in array
            if($users!==false) {
                    foreach($users as $u) {
                            $group_array = json_decode($u->groups, true);
                            $group_array = $this->groups_parse($group_array);

                            if(sizeof($group_array)>0) {
                                    foreach($group_array as $group) {
                                            if(in_array($group_id, $group)) {
                                                    $out[] = $u->id;
                                            }
                                    }
                            }
                    }
            }
            # return
            return isset($out) ? $out : array();
    }
```

10.2.241 Use session_start() Options

WordPress

Use session_start() Options, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

10.2.242 Isset Multiple Arguments

ThinkPHP

Isset Multiple Arguments, in library/think/Request.php:1187.

This may be shortened with isset($sub), $array[$name][$sub])

```
isset($sub) && isset($array[$name][$sub])
```

LiveZilla

Isset Multiple Arguments, in livezilla/_lib/trdp/pchart/class/pDraw.class.php:3852.

This is the equivalent of !(isset($Data[“Series”][$SerieA][“Data”]) && isset($Data[“Series”][$SerieB][“Data”])), and then, !(isset($Data[“Series”][$SerieA][“Data”], $Data[“Series”][$SerieB][“Data”]))

```
!isset($Data["Series"][$SerieA]["Data"]) || !isset($Data["Series"][$SerieB]["Data"])
```

10.2.243 Unitialized Properties

SPIP

Unitialized Properties, in ecrire/public/interfaces.php:584.

The class Critere (Criteria) has no method at all. When using a class as an array, to capture values, one of the advantage of the class is in the default values for the properties. In particular, the last property here, called $not, should be initialized with a false.

```
/**
 * Description d'un critère de boucle
 *
 * Sous-noeud de Boucle
 *
 * @package SPIP\Core\Compilateur\AST
 **/
class Critere {
    /**
     * Type de noeud
     *
     * @var string
     */
    public $type = 'critere';

    /**
     * Opérateur (>, <, >=, IN, ...)
     *
     * @var null|string
     */
    public $op;

    /**
     * Présence d'une négation (truc !op valeur)
     *
     * @var null|string
     */
    public $not;
```

10.2.244 Use List With Foreach

MediaWiki

Use List With Foreach, in includes/parser/LinkHolderArray.php:372.

This foreach reads each element from $entries into entry. $entry, in turn, is written into $pdbk, $title and $displayText for easier reuse. 5 elements are read from $entry, and they could be set in their respective variable in the foreach() with a list call. The only on that can't be set is 'query' which has to be tested.

```
foreach ( $entries as $index => $entry ) {
                            $pdbk = $entry['pdbk'];
                            $title = $entry['title'];
                            $query = isset( $entry['query'] ) ? $entry['query'] : [];
                            $key = "$ns:$index";
                            $searchkey = "<!--LINK'\" $key-->";
                            $displayText = $entry['text'];
                            if ( isset( $entry['selflink'] ) ) {
                                    $replacePairs[$searchkey] =␣
↪Linker::makeSelfLinkObj( $title, $displayText, $query );
                                    continue;
                            }
                            if ( $displayText === '' ) {
                                    $displayText = null;
                            } else {
                                    $displayText = new HtmlArmor( $displayText );
                            }
                            if ( !isset( $colours[$pdbk] ) ) {
                                    $colours[$pdbk] = 'new';
                            }
                            $attribs = [];
                            if ( $colours[$pdbk] == 'new' ) {
                                    $linkCache->addBadLinkObj( $title );
                                    $output->addLink( $title, 0 );
                                    $link = $linkRenderer->makeBrokenLink(
                                            $title, $displayText, $attribs, $query
                                    );
                            } else {
                                    $link = $linkRenderer->makePreloadedLink(
                                            $title, $displayText, $colours[$pdbk],
↪$attribs, $query
                                    );
                            }

                            $replacePairs[$searchkey] = $link;
                    }
```

10.2.245 Empty With Expression

HuMo-Gen

Empty With Expression, in fanchart.php:297.

The test on $pid may be directly done on $treeid[$sosa][0]. The distance between the assignation and the empty() makes it hard to spot.

```
$pid=$treeid[$sosa][0];
                    $birthyr=$treeid[$sosa][1];
                    $deathyr=$treeid[$sosa][4];
                    $fontpx=$fontsize;
                    if($sosa>=16 AND $fandeg==180) { $fontpx=$fontsize-1; }
```

(continues on next page)

(continued from previous page)

```
                    if($sosa>=32 AND $fandeg!=180) { $fontpx=$fontsize-1; }
                    if (!empty($pid)) {
```

10.2.246 Should Use array_filter()

xataface

Should Use array_filter(), in actions/manage_build_index.php:38.

This selection process has three tests : the two first are exclusive, and the third is inclusive. They could fit in one or several closures.

```
$indexable = array();
            foreach ( $tables as $key=>$table ){
                    if ( preg_match('/^dataface__/', $table) ){
                            continue;
                    }
                    if ( preg_match('/^_/', $table) ){
                            continue;
                    }

                    if ( $index->isTableIndexable($table) ){
                            $indexable[] = $table;
                            //unset($tables[$key]);
                    }

            }
```

---

shopware

Should Use array_filter(), in engine/Shopware/Bundle/StoreFrontBundle/Service/Core/VariantCoverService.php:71.

Closure would be the best here, since $covers has to be injected in the array_filter callback.

```
$covers = $this->variantMediaGateway->getCovers(
            $products,
            $context
        );

        $fallback = [];
        foreach ($products as $product) {
            if (!array_key_exists($product->getNumber(), $covers)) {
                $fallback[] = $product;
            }
        }
```

10.2.247 ** For Exponent

Traq

*** For Exponent*, in src/views/layouts/_footer.phtm:5.

pow(1024, 2) could be (1023 ** 2), to convert bytes into Mb.

```
<?=round((microtime(true) - START_TIME), 2); ?>s, <?php echo round((memory_get_peak_
↪usage() - START_MEM) / pow(1024, 2), 3)?>mb
```

TeamPass

*** For Exponent*, in includes/libraries/Authentication/phpseclib/Math/BigInteger.php:286.

pow(2, 62) could also be hard coded with 0x4000000000000000.

```
pow(2, 62)
```

10.2.248 Should Use Math

OpenEMR

Should Use Math, in controllers/C_Prescription.class.php:638.

$pdf->ez['leftMargin'] is now 0.

```
function multiprint_body(& $pdf, $p)
    {
        $pdf->ez['leftMargin'] += $pdf->ez['leftMargin'];
        $pdf->ez['rightMargin'] += $pdf->ez['rightMargin'];
        $d = $this->get_prescription_body_text($p);
        if ($pdf->ezText($d, 10, array(), 1)) {
            $pdf->ez['leftMargin'] -= $pdf->ez['leftMargin'];
            $pdf->ez['rightMargin'] -= $pdf->ez['rightMargin'];
            $this->multiprint_footer($pdf);
            $pdf->ezNewPage();
            $this->multiprint_header($pdf, $p);
```

10.2.249 Could Use Compact

WordPress

Could Use Compact, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

10.2.250 Could Use array_fill_keys

ChurchCRM

Could Use array_fill_keys, in src/ManageEnvelopes.php:107.

There are two initialisations at the same time here : that should make two call to array_fill_keys().

```
foreach ($familyArray as $fam_ID => $fam_Data) {
        $envelopesByFamID[$fam_ID] = 0;
        $envelopesToWrite[$fam_ID] = 0;
    }
```

PhpIPAM

Could Use array_fill_keys, in functions/scripts/merge_databases.php:418.

Even when the initialization is mixed with other operations, it is a good idea to extract it from the loop and give it to array_fill_keys().

```
$arr_new = array();
                            foreach ($arr as $type=>$objects) {
                                    $arr_new[$type] = array();
                                    if(sizeof($objects)>0) {
                                            foreach($objects as $ok=>$object) {
                                                    $arr_new[$type][] = $highest_ids_
↪append[$type] + $object;
                                            }
                                    }
                            }
```

10.2.251 preg_match_all() Flag

FuelCMS

preg_match_all() Flag, in fuel/modules/fuel/helpers/MY_array_helper.php:205.

Using PREG_SET_ORDER will remove the usage of the ‘‘$key‘‘variable.

```
function parse_string_to_array($str)
    {
            preg_match_all('#(\w+)=([\'"])(.*)\2#U', $str, $matches);
            $params = array();
            foreach($matches[1] as $key => $val)
            {
                    if (!empty($matches[3]))
                    {
                            $params[$val] = $matches[3][$key];
                    }
            }
            return $params;
    }
```

10.2.252 Use Count Recursive

WordPress

Use Count Recursive, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

PrestaShop

Use Count Recursive, in controllers/admin/AdminSearchController.php:342.

This could be improved with count() recursive and a array_filter call, to remove empty $list.

```
$nb_results = 0;
            foreach ($this->_list as $list) {
                if ($list != false) {
                    $nb_results += count($list);
                }
            }
```

10.2.253 Should Use Foreach

ExpressionEngine

Should Use Foreach, in system/ee/EllisLab/ExpressionEngine/Service/Model/Query/Builder.php:241.

This code could turn the string into an array, with the explode() function, and use foreach(), instead of calculating the length() initially, and then building the loop.

```
$length = strlen($str);
            $words = array();

            $word = '';
            $quote = '';
            $quoted = FALSE;

            for ($i = 0; $i < $length; $i++)
            {
                    $char = $str[$i];

                    if (($quoted == FALSE && $char == ' ') || ($quoted == TRUE &&
↪$char == $quote))
                    {
                            if (strlen($word) > 2)
                            {
                                    $words[] = $word;
                            }

                            $quoted = FALSE;
                            $quote = '';
                            $word = '';

                            continue;
                    }

                    if ($quoted == FALSE && ($char == ' || $char == ") && ($word === '
↪' || $word == '-'))
                    {
```

(continues on next page)

(continued from previous page)

```
                            $quoted = TRUE;
                            $quote = $char;
                            continue;
                    }

                    $word .= $char;
            }
```

Woocommerce

Should Use Foreach, in includes/libraries/class-wc-eval-math.php:84.

This loops reviews the ‘stack’ and updates its elements. The same loop may leverage foreach and references for more efficient code.

```
$stack_size = count( $stack );
                            for ( $i = 0; $i < $stack_size; $i++ ) { // freeze the␣
↪state of the non-argument variables
                                    $token = $stack[ $i ];
                                    if ( preg_match( '/^[a-z]\w*$/', $token ) and !␣
↪in_array( $token, $args ) ) {
                                            if ( array_key_exists( $token, self::$v )␣
↪) {
                                                    $stack[ $i ] = self::$v[ $token ];
                                            } else {
                                                    return self::trigger( "undefined␣
↪variable '$token' in function definition" );
                                            }
                                    }
                            }
```

10.2.254 Too Many Parameters

WordPress

Too Many Parameters, in wp-admin/includes/misc.php:74.

11 parameters is a lot for a function. Note that it is more than the default configuration, and reported there. This may be configured.

```
/**
 * [identifyUserRights description]
 * @param  string $groupesVisiblesUser  [description]
 * @param  string $groupesInterditsUser [description]
 * @param  string $isAdmin              [description]
 * @param  string $idFonctions          [description]
 * @return string                       [description]
 */
function identifyUserRights(
    $groupesVisiblesUser,
    $groupesInterditsUser,
    $isAdmin,
```

(continues on next page)

(continued from previous page)

```
    $idFonctions,
    $server,
    $user,
    $pass,
    $database,
    $port,
    $encoding,
    $SETTINGS
) {
```

ChurchCRM

Too Many Parameters, in src/Reports/ReminderReport.php:192.

10 parameters is a lot for a function. Here, we may also identify a family (ID, Name), and a full address (Address1, Address2, State, Zip, Country), which may be turned into an object.

```
public function StartNewPage($fam_ID, $fam_Name, $fam_Address1, $fam_Address2, $fam_
↪City, $fam_State, $fam_Zip, $fam_Country, $fundOnlyString, $iFYID)
{
```

10.2.255 Should Preprocess Chr()

phpadsnew

Should Preprocess Chr(), in phpAdsNew-2.0/adview.php:302.

Each call to chr() may be done before. First, chr() may be replace with the hexadecimal sequence "0x3B"; Secondly, 0x3b is a rather long replacement for a simple semi-colon. The whole pragraph could be stored in a separate file, for easier modifications.

```
echo chr(0x47).chr(0x49).chr(0x46).chr(0x38).chr(0x39).chr(0x61).chr(0x01).chr(0x00).
                 chr(0x01).chr(0x00).chr(0x80).chr(0x00).chr(0x00).chr(0x04).
↪chr(0x02).chr(0x04).
                     chr(0x00).chr(0x00).chr(0x00).chr(0x21).chr(0xF9).chr(0x04).
↪chr(0x01).chr(0x00).
                 chr(0x00).chr(0x00).chr(0x00).chr(0x2C).chr(0x00).chr(0x00).
↪chr(0x00).chr(0x00).
                 chr(0x01).chr(0x00).chr(0x01).chr(0x00).chr(0x00).chr(0x02).
↪chr(0x02).chr(0x44).
                 chr(0x01).chr(0x00).chr(0x3B);
```

10.2.256 Drop Substr Last Arg

SuiteCrm

Drop Substr Last Arg, in modules/UpgradeWizard/uw_utils.php:2422.

substr() is even trying to go beyond the end of the string.

```
substr($relativeFile, 1, strlen($relativeFile))
```

Tine20

Drop Substr Last Arg, in tine20/Calendar/Frontend/Cli.php:95.

Omitting the last character would yield the same result.

```
substr($opt, 18, strlen($opt))
```

10.2.257 Possible Increment

Zurmo

Possible Increment, in app/protected/modules/workflows/utils/SavedWorkflowsUtil.php:196.

There are suspicious extra spaces around the +, that give the hint that there used to be something else, like a constant, there. From the name of the methods, it seems that this code was refactored from an addition to a simple method call.

```
$timeStamp =  + $workflow->getTimeTrigger()->resolveNewTimeStampForDuration(time());
```

MediaWiki

Possible Increment, in includes/filerepo/file/LocalFile.php:613.

That is a useless assignation, except for the transtyping to integer that PHP does silently. May be that should be a +=, or completely dropped.

```
$decoded[$field] = +$decoded[$field]
```

10.2.258 One If Is Sufficient

Tikiwiki

One If Is Sufficient, in lib/wiki-plugins/wikiplugin_trade.php:152.

empty($params['inputtitle']) should have priority over $params['wanted'] == 'n'.

```
if ($params['wanted'] == 'n') {
            if (empty($params['inputtitle'])) {
                    $params['inputtitle'] = 'Payment of %0 %1 from user %2 to %3';
            }
    } else {
            if (empty($params['inputtitle'])) {
                    $params['inputtitle'] = 'Request payment of %0 %1 to user %2 from
↪%3';
            }
    }
```

10.2.259 Could Use array_unique

Dolibarr

Could Use array_unique, in htdocs/includes/restler/framework/Luracast/Restler/Format/XmlFormat.php:250.

This loop has two distinct operations : the first collect keys and keep them unique. A combinaison of array_keys() and array_unique() would do that job, while saving the in_array() lookup, and the configuration check with 'static::$importSettingsFromXml'. The second operation is distinct, and could be done with array_map().

```
$attributes = $xml->attributes();
            foreach ($attributes as $key => $value) {
                if (static::$importSettingsFromXml
                    && !in_array($key, static::$attributeNames)
                ) {
                    static::$attributeNames[] = $key;
                }
                $r[$key] = static::setType((string)$value);
            }
```

OpenEMR

Could Use array_unique, in gacl/gacl_api.class.php:441:441.

This loop is quite complex : it collects $aro_value in $acl_array['aro'][$aro_section_value], but also creates the array in $acl_array['aro'][$aro_section_value], and report errors in the debug log. array_unique() could replace the collection, while the debug would have to be done somewhere else.

```
foreach ($aro_value_array as $aro_value) {
                                    if ( count($acl_array['aro'][$aro_section_value])␣
↪!= 0 ) {
                                            if (!in_array($aro_value, $acl_array['aro
↪'][$aro_section_value])) {
                                                    $this->debug_text("append_acl():␣
↪ARO Section Value: $aro_section_value ARO VALUE: $aro_value");
                                                    $acl_array['aro'][$aro_section_
↪value][] = $aro_value;
                                                    $update=1;
                                            } else {
                                                    $this->debug_text("append_acl():␣
↪Duplicate ARO, ignoring... ");
                                            }
                                    } else { //Array is empty so add this aro value.
                                            $acl_array['aro'][$aro_section_value][] =
↪$aro_value;
                                            $update = 1;
                                    }
                            }
```

10.2.260 Too Many Children

Typo3

Too Many Children, in typo3/sysext/backend/Classes/Form/AbstractNode.php:26.

More than 15 children for this class : 15 is the default configuration.

```
abstract class AbstractNode implements NodeInterface, LoggerAwareInterface {
```

Woocommerce

Too Many Children, in includes/abstracts/abstract-wc-rest-controller.php:30.

This class is extended 22 times, more than the default configuration of 15.

```
class WC_REST_Controller extends WP_REST_Controller {
```

10.2.261 Should Use Operator

Zencart

Should Use Operator, in includes/modules/payment/paypal/paypal_curl.php:378.

Here, $options is merged with $values if it is an array. If it is not an array, it is probably a null value, and may be ignored. Adding a 'array' typehint will strengthen the code an catch situations where TransactionSearch() is called with a string, leading to clearer code.

```
function TransactionSearch($startdate, $txnID = '', $email = '', $options) {
    // several lines of code, no mention of $options
      if (is_array($options)) $values = array_merge($values, $options);
    }
    return $this->_request($values, 'TransactionSearch');
  }
```

SugarCrm

Should Use Operator, in include/utils.php:2093:464.

$override should an an array : if not, it is actually set by default to empty array. Here, a typehint with a default value of 'array()' would offset the parameter validation to the calling method.

```
function sugar_config_union( $default, $override ){
    // a little different then array_merge and array_merge_recursive.  we want
    // the second array to override the first array if the same value exists,
    // otherwise merge the unique keys.  it handles arrays of arrays recursively
    // might be suitable for a generic array_union
    if( !is_array( $override ) ){
            $override = array();
    }
    foreach( $default as $key => $value ){
            if( !array_key_exists($key, $override) ){
                    $override[$key] = $value;
            }
            else if( is_array( $key ) ){
                    $override[$key] = sugar_config_union( $value, $override[$key] );
```

(continues on next page)

(continued from previous page)

```
            }
    }
    return( $override );
}
```

10.2.262 Could Be Static Closure

Piwigo

Could Be Static Closure, in include/ws_core.inc.php:620.

The closure function($m) makes no usage of the current object : using static prevents $this to be forwarded with the closure.

```
/**
   * WS reflection method implementation: lists all available methods
   */
  static function ws_getMethodList($params, &$service)
  {
    $methods = array_filter($service->_methods,
      function($m) { return empty($m["options"]["hidden"]) || !$m["options"]["hidden
↪"];} );
    return array('methods' => new PwgNamedArray( array_keys($methods),'method' ) );
  }
```

10.2.263 Add Default Value

Zurmo

Add Default Value, in wp-admin/includes/misc.php:74.

Default values may be a literal (1, 'abc', …), or a constant : global or class. Here, MissionsListConfigurationForm::LIST_TYPE_AVAILABLE may be used directly in the signature of the method

```
public function getMetadataFilteredByOption($option)
        {
            if ($option == null)
            {
                $option = MissionsListConfigurationForm::LIST_TYPE_AVAILABLE;
            }
```

Typo3

Add Default Value, in typo3/sysext/indexed_search/Classes/FileContentParser.php:821.

$extension could get a default value to handle default situations : for example, a file is htm format by default, unless better known. Also, the if/then structure could get a 'else' clause, to handle unknown situations : those are situations where the extension is provided but not known, in particular when the icon is missing in the storage folder.

```
public function getIcon($extension)
    {
        if ($extension === 'htm') {
            $extension = 'html';
        } elseif ($extension === 'jpeg') {
            $extension = 'jpg';
        }
        return 'EXT:indexed_search/Resources/Public/Icons/FileTypes/' . $extension .
↪'.gif';
    }
```

10.2.264 Named Regex

Phinx

Named Regex, in src/Phinx/Util/Util.php:127.

$matches[1] could be renamed by $matches['filename'], if the capturing subpattern was named 'filename'.

```
const MIGRATION_FILE_NAME_PATTERN = '/^\d+_([\w_]+).php$/i';
//.... More code with class definition
    public static function mapFileNameToClassName($fileName)
    {
        $matches = [];
        if (preg_match(static::MIGRATION_FILE_NAME_PATTERN, $fileName, $matches)) {
            $fileName = $matches[1];
        }

        return str_replace(' ', '', ucwords(str_replace('_', ' ', $fileName)));
    }
```

---

shopware

Named Regex, in engine/Library/Enlight/Components/Snippet/Resource.php:207.

$_match[3] is actually extracted two preg_match() before : by the time we read its usage, the first regex has been forgotten. A named subpattern would be useful here to remember what was captured.

```
if (!preg_match("!(.?)(name=)(.*?)(?=(\s|$))!", $_block_args, $_match) && empty($_
↪block_default)) {
                throw new SmartyException('"' . $_block_tag . '" missing name␣
↪attribute');
            }
            $_block_force = (bool) preg_match('#[\s]force#', $_block_args);
            $_block_json = (bool) preg_match('#[\s]json=["\']true["\']\W#', $_block_
↪args);
            $_block_name = !empty($_match[3]) ? trim($_match[3], '\'"') : $_block_
↪default;
```

10.2.265 Could Use Try

Mautic

Could Use Try, in app/bundles/StageBundle/Controller/StageController.php:78.

$limit is read as a session variable or a default value. There are no check here that $limit is not null, before using it in a division. It is easy to imagine this is done elsewhere, yet a try/catch could help intercept unwanted situations.

```
//set limits
        $limit = $this->get('session')->get(
            'mautic.stage.limit',
            $this->coreParametersHelper->getParameter('default_pagelimit')
        );
/... Code where $limit is read but not modified /
        $count = count($stages);
        if ($count && $count < ($start + 1)) {
            $lastPage = ($count === 1) ? 1 : (ceil($count / $limit)) ?: 1;
```

10.2.266 Use Basename Suffix

NextCloud

Use Basename Suffix, in lib/private/URLGenerator.php:176.

This code removes the 4 last letters from the images. It may be ‘png’, ‘jpg’ or ‘txt’.

```
substr(basename($image), 0, -4)
```

Dolibarr

Use Basename Suffix, in htdocs/core/website.inc.php:42.

The extension ‘.tpl.php’ is dropped from the file name, unless it appears somewhere else in the $websitepagefile variable.

```
str_replace(array('.tpl.php', 'page'), array('', ''), basename($websitepagefile))
```

10.2.267 Don’t Loop On Yield

Dolibarr

Don’t Loop On Yield, in htdocs/includes/sabre/sabre/dav/lib/DAV/Server.php:912.

Yield from is a straight replacement here.

```
if (($newDepth === self::DEPTH_INFINITY || $newDepth >= 1) && $childNode instanceof␣
↪ICollection) {
    foreach ($this->generatePathNodes($subPropFind) as $subItem) {
        yield $subItem;
    }
}
```

Tikiwiki

Don't Loop On Yield, in lib/goal/goallib.php:944.

The replacement with yield from``is not straigthforward here. Yield is only called when $user hasn't been ``$done : this is a unicity check. So, the double loop may produce a fully merged array, that may be reduced further by array_unique(). The final array, then, can be used with yield from.

```
$done = [];

foreach ($goal['eligible'] as $groupName) {
    foreach ($userlib->get_group_users($groupName) as $user) {
            if (! isset($done[$user])) {
                    yield ['user' => $user, 'group' => null];
                    $done[$user] = true;
            }
    }
}
```

10.2.268 Multiple Usage Of Same Trait

NextCloud

Multiple Usage Of Same Trait, in build/integration/features/bootstrap/WebDav.php:41.

WebDav uses Sharing, and Sharing uses Webdav. Once using the other is sufficient.

```
trait WebDav {
    use Sharing;

}
//Trait Sharing is in /build/integration/features/bootstrap/Sharing.php:36
```

10.2.269 Function Subscripting, Old Style

OpenConf

Function Subscripting, Old Style, in openconf/include.php:1469.

Here, $advocateid may be directly read from ocsql_fetch_assoc(), although, checking for the existence of 'advocateid' before accessing it would make the code more robust

```
$advocateid = false;
    if (isset($GLOBALS['OC_configAR']['OC_paperAdvocates']) && $GLOBALS['OC_configAR
↪']['OC_paperAdvocates']) {
            $ar = ocsql_query(SELECT `advocateid` FROM ` . OCC_TABLE_PAPERADVOCATE .␣
↪` WHERE `paperid`=' . safeSQLstr($pid) . ') or err('Unable to retrieve advocate');
            if (ocsql_num_rows($ar) == 1) {
                    $al = ocsql_fetch_assoc($ar);
                    $advocateid = $al['advocateid'];
            }
    }
```

10.2.270 No Class As Typehint

Vanilla

No Class As Typehint, in library/Vanilla/Formatting/Formats/RichFormat.php:51.

All three typehints are based on classes. When Parser or Renderer are changed, for testing, versioning or moduling reasons, they must subclass the original class.

```
public function __construct(Quill\Parser $parser, Quill\Renderer $renderer,␣
↪Quill\Filterer $filterer) {
        $this->parser = $parser;
        $this->renderer = $renderer;
        $this->filterer = $filterer;
    }
```

---

phpMyAdmin

No Class As Typehint, in libraries/classes/CreateAddField.php:29.

Although the class is named 'DatabaseInterface', it is a class.

```
public function __construct(DatabaseInterface $dbi)
    {
        $this->dbi = $dbi;
    }
```

10.2.271 Argument Should Be Typehinted

Dolphin

Argument Should Be Typehinted, in Dolphin-v.7.3.5/plugins/intervention-image/Intervention/Image/Gd/Commands/WidenCommand.php

This closures make immediate use of the $constraint argument, and calls its method aspectRatio. No check is made on this argument, and it may easily be mistaken with another class, or a null. Adding a typehint here will ensure a more verbose development error and help detect misuse of the closure.

```
$this->arguments[2] = function ($constraint) use ($additionalConstraints) {
            $constraint->aspectRatio();
            if(is_callable($additionalConstraints))
                $additionalConstraints($constraint);
        };
```

Mautic

Argument Should Be Typehinted, in app/bundles/PluginBundle/Helper/IntegrationHelper.php:374.

This piece of code inside a 275 lines method. Besides, there are 11 classes that offer a 'getPriority' method, although $returnServices could help to semantically reduce the number of possible classes. Here, typehints on $a and $b help using the wrong kind of object.

```
if (empty($alphabetical)) {
            // Sort by priority
            uasort($returnServices, function ($a, $b) {
                $aP = (int) $a->getPriority();
                $bP = (int) $b->getPriority();

                if ($aP === $bP) {
                    return 0;
                }

                return ($aP < $bP) ? -1 : 1;
            });
```

CHAPTER 11

Reports

There are several reports that may be extracted from Exakat :

- *Ambassador*
- *BeautyCanon*
- *ClassReview*
- *Classes dependendies HTML*
- *Clustergrammer*
- *Code Flower*
- *Code Sniffer*
- *Composer*
- *Dependency Wheel*
- *Diplomat*
- *Exakatyaml*
- *File dependendies*
- *File dependendies HTML*
- *History*
- *Inventories*
- *Json*
- *Marmelab*
- *Meters*
- *Migration74*
- *Migration80*
- *None*

- *Owasp*
- *Perfile*
- *PhpCompilation*
- *PhpConfiguration*
- *Phpcity*
- *Phpcsfixer*
- *PlantUml*
- *RadwellCode*
- *Rector*
- Sarb
- *Sarif*
- *SimpleTable*
- *Stats*
- *Stubs*
- *StubsJson*
- *Text*
- *Top10*
- *Topology Order*
- *TypeChecks*
- *TypeSuggestion*
- *Uml*
- *Xml*
- *Yaml*

11.1 Configuring a report before the audit

By default, Exakat builds the 'Ambassador' report for any project. If you want another report, or want to ignore the build of Ambassador, configure it before running the audit.

To do so, open the *projects/<project>/config.ini* file, and mention the list of report like that :

```
project_reports[] = 'Owasp';
project_reports[] = 'Weekly';
```

By configuring the reports before the audit, Exakat processes only the needed analysis, and produces all the reports for each audit.

11.2 Generating a report after the audit

If you have run an audit, but wants to extract another report for a piece of code, you can use the following command :

> *php exakat.phar report -p <project> -format <format> -file <filename>*

Where <format> is one of the format listed in the following section, and <filename> is the target file.

Note that some format requires some specific audits to be run : they will fail if those results are not available. Then, run the audit again, and mention the desired audit in the configuration.

11.3 Common behavior

Default format is Text. Each report has a default filename, that may be configured with the -file option. Each report adds a file extension to the provided filename.

A special value for -file is 'stdout'. Some formats may be output to stdout, such as Text or Json. Not all format are accepting that value : some format, like Ambassador or Sqlite, may only be written to directories.

Each report is stored in its <project> folder, under the requested name.

Reports may be generated at any time, during execution of the analysis (partial results) or later, even if another audit is running.

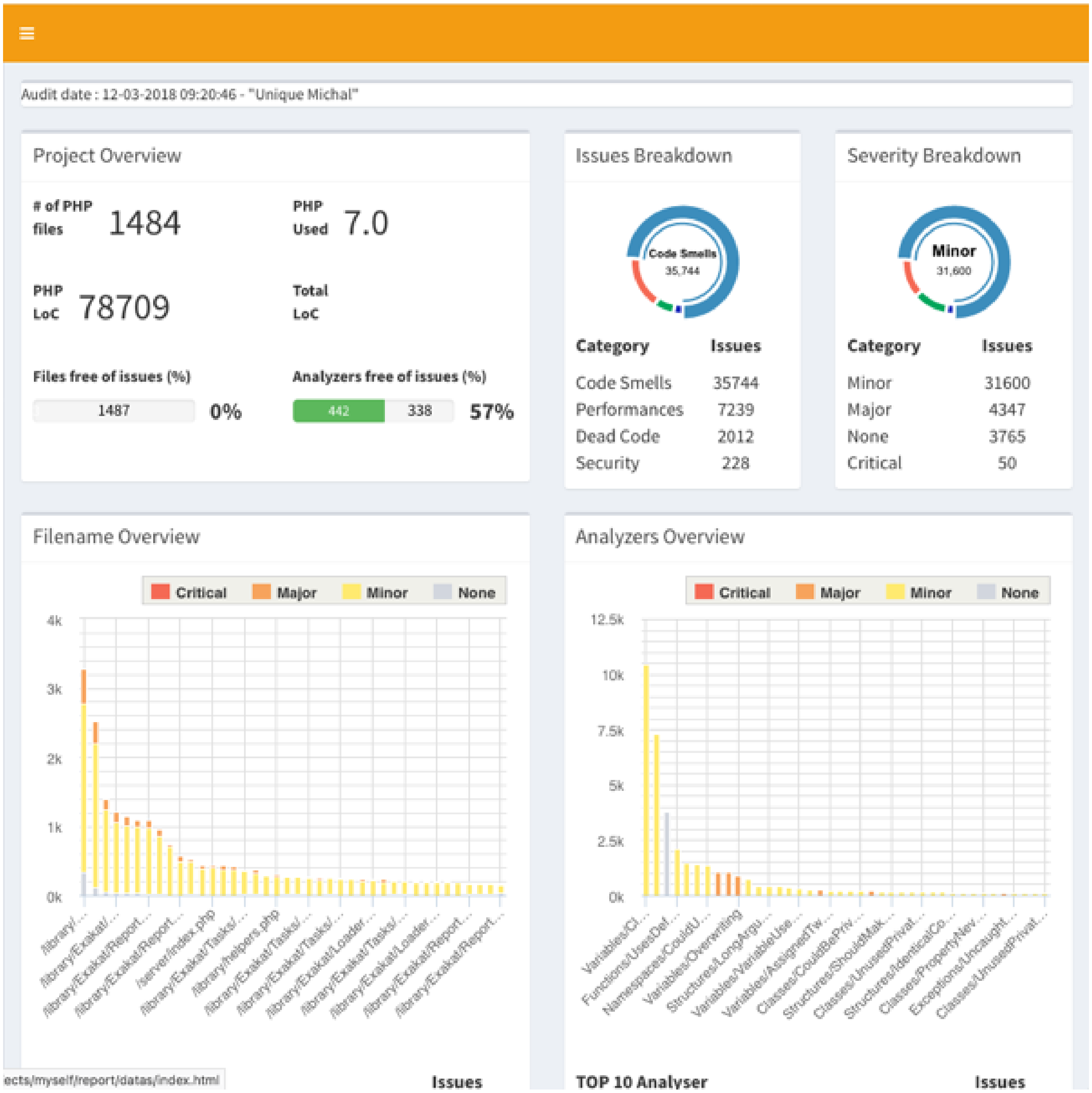
11.4 Reports descriptions

11.4.1 Ambassador

Ambassador is the most complete Exakat report. It used to be the default report, until Exakat 1.7.0

The Ambassador report includes :

- Full configuration for the audit
- Full documentation of the analysis
- All results, searchable and browsable by file and analysis
- **Extra reports for**
    - Minor versions compatibility
    - PHP Directive usage
    - PHP compilation recommendations
    - Error messages list
    - List of processed files



Audit date : 12-03-2018 09:20:46 - "Unique Michal"

Project Overview

# of PHP files 1484

PHP Used 7.0

PHP LoC 78709

Total LoC

Files free of issues (%) 1487 0%

Analyzers free of issues (%) 442 338 57%

Issues Breakdown

Category	Issues
Code Smells	35744
Performances	7239
Dead Code	2012
Security	228

Severity Breakdown

Category	Issues
Minor	31600
Major	4347
None	3765
Critical	50

Filename Overview

Analyzers Overview

ects/myself/report/datas/index.html

Issues

TOP 10 Analyser

Issues

Ambassador includes the report from 3 other reports : PhpCompilation, PhpConfiguration, Stats.

Ambassador is a HTML report format.

Ambassador depends on the following 20 themes : CompatibilityPHP53, CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72, CompatibilityPHP73, CompatibilityPHP74, CompatibilityPHP80, Analyze, Preferences, Inventory, Performances, Appinfo, Appcontent, Dead code, Security, Suggestions, Custom.

11.4.2 BeautyCanon

The Beauty Canon report lists all rules that report no issues.

The Beauty Canon report displays one result per line. This report lists all issues in the provided ruleset that are reporting no error.

The title of the analysis is listed on the left, and the analysis short name is listed on the right, for further documentation.

This analysis uses Analysis as default rule. It may otherwise parametered with the -T option.

Compare Hash *Compare Hash* Configure Extract *Configure Extract* Dynamic Library Loading *Dynamic Library Loading* Encoded Simple Letters *Encoded Simple Letters* Indirect Injection *Indirect Injection* Integer Conversion *Integer Conversion* Minus One On Error *Minus One On Error* Mkdir Default *Mkdir Default* No ENT_IGNORE *No ENT_IGNORE* No Hardcoded Hash *No Hardcoded Hash* No Hardcoded Ip *No Hardcoded Ip* No Hardcoded Port *No Hardcoded Port*

```
Compare Hash                                                            Security/
↪CompareHash
Configure Extract                                                       Security/
↪ConfigureExtract
Dynamic Library Loading                                                 Security/
↪DynamicDl
Encoded Simple Letters                                                  Security/
↪EncodedLetters
Indirect Injection                                                      Security/
↪IndirectInjection
Integer Conversion                                                      Security/
↪IntegerConversion
Minus One On Error                                                      Security/
↪MinusOneOnError
Mkdir Default                                                           Security/
↪MkdirDefault
No ENT_IGNORE                                                           Security/
↪NoEntIgnore
No Hardcoded Hash                                                       Structures/
↪NoHardcodedHash
No Hardcoded Ip                                                         Structures/
↪NoHardcodedIp
No Hardcoded Port                                                       Structures/
↪NoHardcodedPort
```

BeautyCanon is a Text report format.

BeautyCanon accepts any arbitrary list of results.

11.4.3 ClassReview

The ClassReview report focuses on reviewing classes, traits and interfaces.

The ClassReview report focuses on good code hygiene for classes, interfaces and traits.

It checks the internal structure of classes, and suggest visibility, typehint updates.

ClassReview is a HTML report format.

ClassReview depends on the following theme : ClassReview.

11.4.4 Classes dependendies HTML

This reports displays the class dependencies, based on definition usages.

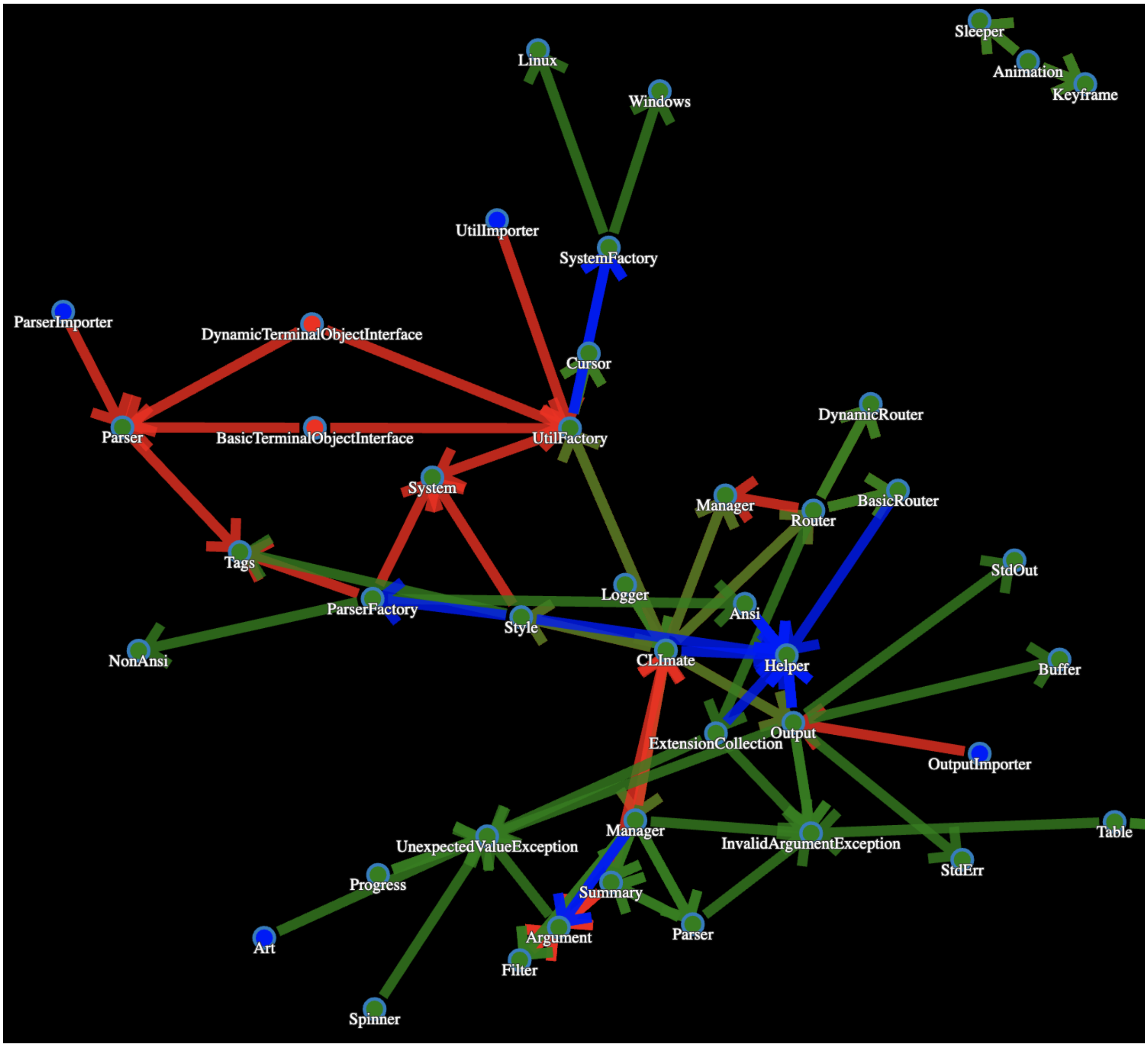
This report displays all dependencies between classes, interfaces and traits. A class (or interface or trait) depends on another class (or interface or trait) when it makes usage of one of its definitions : extends, implements, use, and static calls.

For example, *A* depends on *B*, because *A* extends *B*.

The resulting diagram is in HTML file, which is readable with most browsers, from a web server.

Warning : for browser security reasons, the report will NOT load as a local file. It needs to be served by an HTTP server, so all resources are correctly located.

Warning : large applications (> 1000 classes) will require a lot of resources to open.



Classes dependendies HTML is a HTML report format.

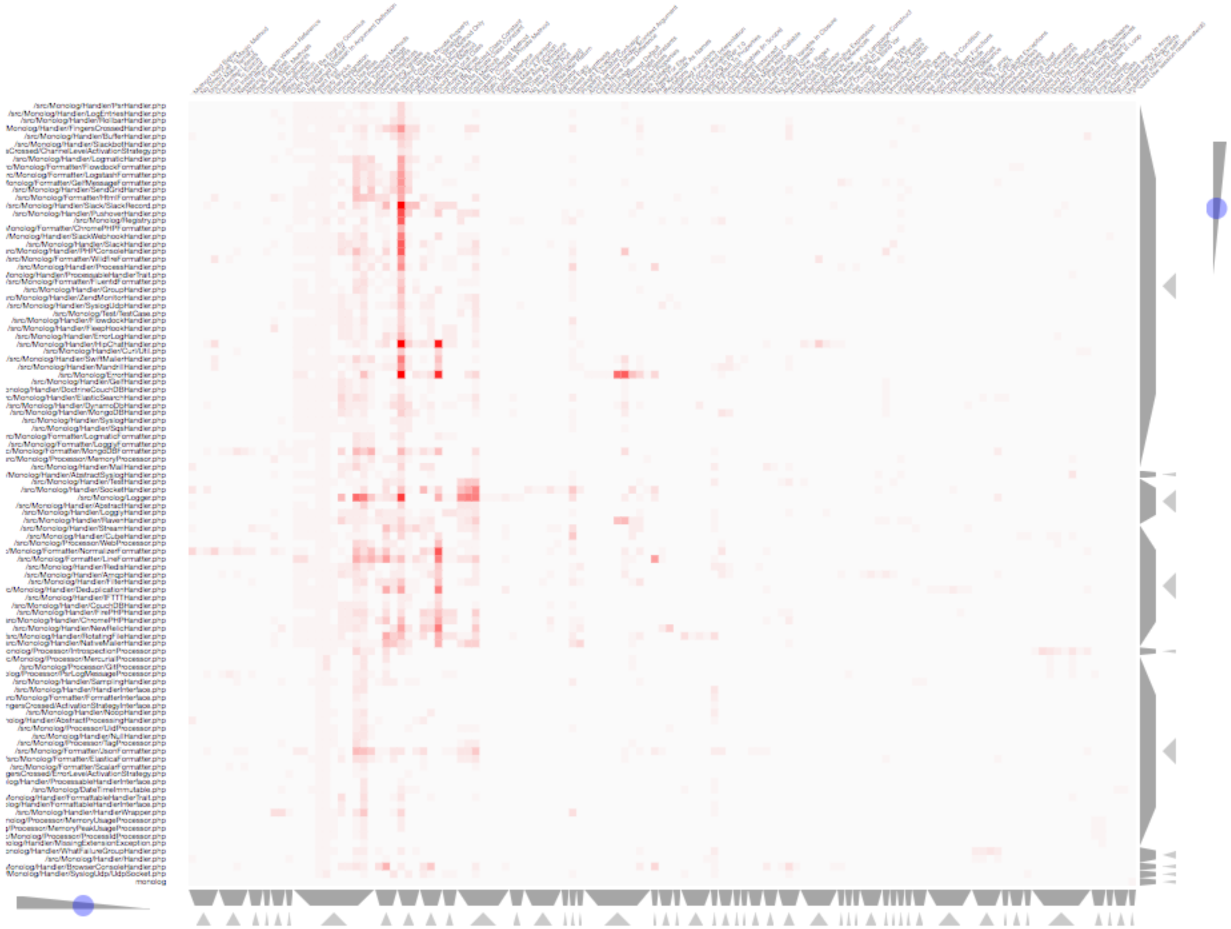
Classes dependendies HTML doesn't depend on themes.

11.4.5 Clustergrammer

The Clustergrammar report format data for a clustergrammer diagram.

Clustergrammer is a visualisation tool that may be found online. After generation of this report, a TEXT file is available in the project directory. Upload it on [http://amp.pharm.mssm.edu/clustergrammer/{]}(http://amp.pharm.mssm.edu/clustergrammer/) to visualize it.

See a live report here : [Clustergrammer](http://amp.pharm.mssm.edu/clustergrammer/viz_sim_mats/5a8d41bf3a82d32a9dacddd9/clustergrammer.txt).



Clustergrammer is a TEXT report format.

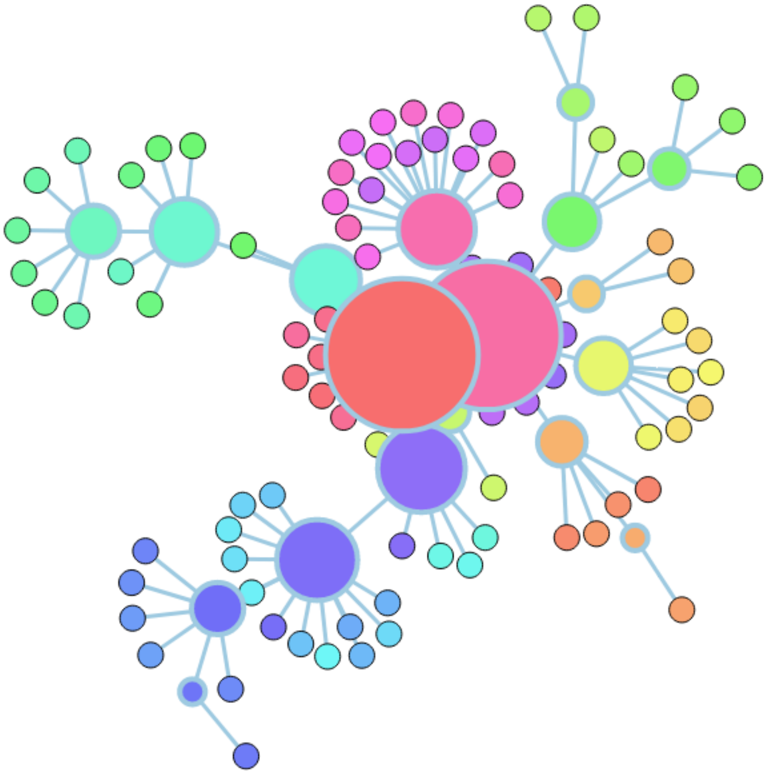
Clustergrammer doesn't depend on themes.

11.4.6 Code Flower

The Code Flower represents hierarchies in a code source.

Codeflower is a javascript visualization of the code. It is based on Francois Zaninotto's [CodeFlower Source code visualization](http://www.redotheweb.com/CodeFlower/).

It represents : + Class hierarchy + Namespace hierarchy + Inclusion



Code Flower is a HTML report format.

Code Flower doesn't depend on themes.

11.4.7 Code Sniffer

The CodeSniffer report exports in the CodeSniffer format.

This format reports analysis using the Codesniffer's result format.

See also [Code Sniffer Report](https://github.com/squizlabs/PHP_CodeSniffer/wiki/Reporting).

```
FILE : /Path/To/View/The/File.php
--------------------------------------------------------------------------------
FOUND 3 ISSUES AFFECTING 3 LINES
```

(continues on next page)

(continued from previous page)

```
--------------------------------------------------------------------------------
 32 | MINOR | Could Use Alias
 41 | MINOR | Could Make A Function
 43 | MINOR | Could Make A Function
--------------------------------------------------------------------------------
```

Code Sniffer is a TEXT report format.

Code Sniffer accepts any arbitrary list of results.

11.4.8 Composer

The Composer report provide elements for the require attribute in the composer.json.

It helps documenting the composer.json, by providing more information, extracted from the code.

This report makes a copy then updates the composer.json, if available. It creates a totally new composer.json if the latter is not available.

It is recommended to review manually the results of the suggested composer.json before using it.

```
Name,File,Line
0,/features/bootstrap/FeatureContext.php,61
10000,/features/bootstrap/FeatureContext.php,61
777,/features/bootstrap/FeatureContext.php,63
20,/features/bootstrap/FeatureContext.php,73
0,/features/bootstrap/FeatureContext.php,334
0,/features/bootstrap/FeatureContext.php,339
0,/features/bootstrap/FeatureContext.php,344
0,/features/bootstrap/FeatureContext.php,362
0,/features/bootstrap/FeatureContext.php,366
0,/features/bootstrap/FeatureContext.php,368
0,/features/bootstrap/FeatureContext.php,372
777,/features/bootstrap/FeatureContext.php,423
777,/features/bootstrap/FeatureContext.php,431
0,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,68
1,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,69
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,84
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,150
```

Composer is a JSON report format.

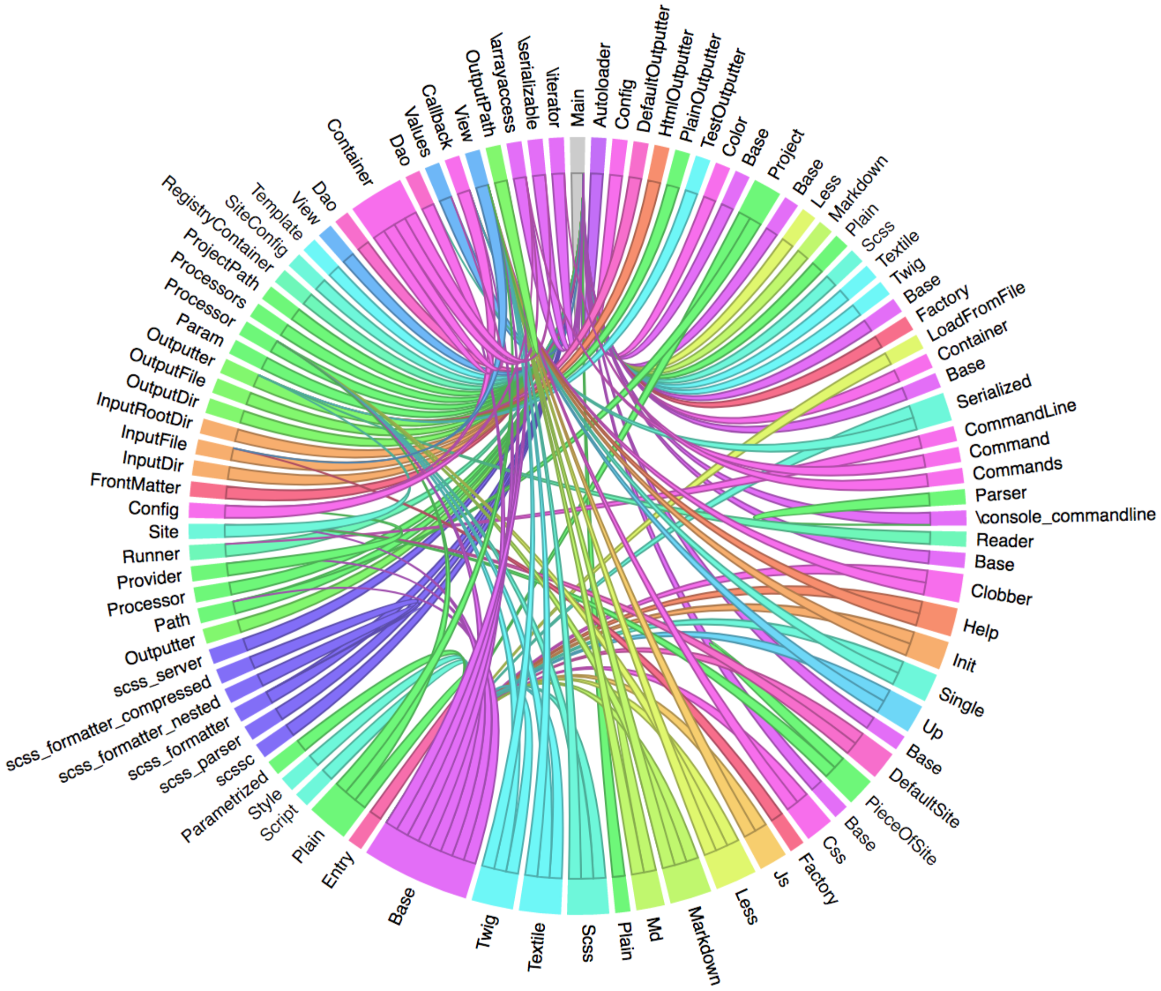
Composer depends on the following theme : Appinfo.

11.4.9 Dependency Wheel

The DependencyWheel represents dependencies in a code source.

Dependency Wheel is a javascript visualization of the classes dependencies in the code. Every class, interface and trait are represented as a circle, and every relation between the classes are represented by a link between them, inside the circle.

It is based on Francois Zaninotto’s **‘DependencyWheel <http://fzaninotto.github.com/DependencyWheel‘_** and the d3.js.



Dependency Wheel is a HTML report format.

Dependency Wheel doesn’t depend on themes.

11.4.10 Diplomat

The Diplomat is the default human readable report.

The Diplomat report is the default report since Exakat 1.7.0. It is a light version of the Ambassador report, and uses a shorter list of analysis.

```
Name,File,Line
0,/features/bootstrap/FeatureContext.php,61
10000,/features/bootstrap/FeatureContext.php,61
777,/features/bootstrap/FeatureContext.php,63
20,/features/bootstrap/FeatureContext.php,73
0,/features/bootstrap/FeatureContext.php,334
0,/features/bootstrap/FeatureContext.php,339
0,/features/bootstrap/FeatureContext.php,344
0,/features/bootstrap/FeatureContext.php,362
```

(continues on next page)

(continued from previous page)

```
0,/features/bootstrap/FeatureContext.php,366
0,/features/bootstrap/FeatureContext.php,368
0,/features/bootstrap/FeatureContext.php,372
777,/features/bootstrap/FeatureContext.php,423
777,/features/bootstrap/FeatureContext.php,431
0,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,68
1,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,69
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,84
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,150
```

Diplomat is a HTML report format.

Diplomat depends on the following 15 themes : CompatibilityPHP53, CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72, CompatibilityPHP73, CompatibilityPHP74, CompatibilityPHP80, Top10, Preferences, Appinfo, Appcontent, Suggestions.

11.4.11 Exakatyaml

Builds a list of ruleset, based on the number of issues from the previous audit.

Exakatyaml helpls with the configuration of exakat in a CI. It builds a list of ruleset, based on the number of issues from the previous audit.

Continuous Integration require steps that yield no issues. This is good for analysis that yield no results : in a word, all analysis that are currently clean should be in the CI. That way, any return will be monitored.

On the other hand, other analysis that currently yield issues needs to be fully cleaned before usage.

```
project: my_project
project_name: my_project
project_themes: {  }
project_reports:
    - Ambassador
rulesets:
    ruleset_0: # 0 errors found
         "Accessing Private":                                 Classes/AccessPrivate
         "Adding Zero":                                       Structures/AddZero
         "Aliases Usage":                                     Functions/AliasesUsage
         "Already Parents Interface":                         Interfaces/
↪AlreadyParentsInterface
         "Already Parents Trait":                             Traits/
↪AlreadyParentsTrait
         "Altering Foreach Without Reference":                Structures/
↪AlteringForeachWithoutReference
         "Alternative Syntax Consistence":                    Structures/
↪AlternativeConsistenceByFile
         "Always Positive Comparison":                        Structures/NeverNegative
# Other results here
    ruleset_1: # 1 errors found
         "Constant Class":                                    Classes/ConstantClass
         "Could Be Abstract Class":                           Classes/
↪CouldBeAbstractClass
         "Dependant Trait":                                   Traits/DependantTrait
         "Double Instructions":                               Structures/
↪DoubleInstruction
# Other results here
```

(continues on next page)

(continued from previous page)

```
    ruleset_2: # 2 errors found
        "Always Anchor Regex":                                Security/AnchorRegex
        "Forgotten Interface":                                Interfaces/
↪CouldUseInterface
# Other results here
    ruleset_3: # 3 errors found
        "@ Operator":                                         Structures/Noscream
        "Indices Are Int Or String":                          Structures/
↪IndicesAreIntOrString
        "Modernize Empty With Expression":                    Structures/ModernEmpty
        "Property Variable Confusion":                        Structures/
↪PropertyVariableConfusion
# Other results here
    ruleset_4: # 4 errors found
        "Buried Assignation":                                 Structures/
↪BuriedAssignation
        "Identical Consecutive Expression":                   Structures/
↪IdenticalConsecutive
# Other results here
    ruleset_122: # 122 errors found
        "Method Could Be Static":                             Classes/CouldBeStatic
```

```
project: page_manager
project_name: drupal_page_manager
project_themes: {  }
project_reports:
    - Ambassador
rulesets:
    ruleset_0: # 0 errors found
        "$HTTP_RAW_POST_DATA Usage":                          Php/RawPostDataUsage
        "$this Belongs To Classes Or Traits":                 Classes/ThisIsForClasses
        "$this Is Not An Array":                              Classes/ThisIsNotAnArray
        "$this Is Not For Static Methods":                    Classes/
↪ThisIsNotForStatic
        "Abstract Or Implements":                             Classes/
↪AbstractOrImplements
        "Access Protected Structures":                        Classes/AccessProtected
        "Accessing Private":                                  Classes/AccessPrivate
        "Adding Zero":                                        Structures/AddZero
        "Aliases Usage":                                      Functions/AliasesUsage
        "Already Parents Interface":                          Interfaces/
↪AlreadyParentsInterface
        "Already Parents Trait":                              Traits/
↪AlreadyParentsTrait
        "Altering Foreach Without Reference":                 Structures/
↪AlteringForeachWithoutReference
        "Alternative Syntax Consistence":                     Structures/
↪AlternativeConsistenceByFile
        "Always Positive Comparison":                         Structures/NeverNegative
        "Ambiguous Array Index":                              Arrays/AmbiguousKeys
        "Ambiguous Static":                                   Classes/AmbiguousStatic
        "Ambiguous Visibilities":                             Classes/
↪AmbiguousVisibilities
        "Anonymous Classes":                                  Classes/Anonymous
        "Assert Function Is Reserved":                        Php/
↪AssertFunctionIsReserved
```

(continues on next page)

(continued from previous page)

```
        "Assign And Compare":                                Structures/
↪AssigneAndCompare
        "Assign Default To Properties":                      Classes/MakeDefault
        "Assign With And":                                   Php/AssignAnd
        "Assigned Twice":                                    Variables/
↪AssignedTwiceOrMore
        "Avoid Parenthesis":                                 Structures/
↪PrintWithoutParenthesis
        "Avoid Those Hash Functions":                        Security/
↪AvoidThoseCrypto
        "Avoid Using stdClass":                              Php/UseStdclass
        "Avoid get_class()":                                 Structures/UseInstanceof
        "Avoid option arrays in constructors":               Classes/
↪AvoidOptionArrays
        "Avoid set_error_handler $context Argument":         Php/
↪AvoidSetErrorHandlerContextArg
        "Avoid sleep()/usleep()":                            Security/NoSleep
        "Bad Constants Names":                               Constants/
↪BadConstantnames
        "Callback Needs Return":                             Functions/
↪CallbackNeedsReturn
        "Can't Count Non-Countable":                         Structures/
↪CanCountNonCountable
        "Can't Extend Final":                                Classes/CantExtendFinal
        "Can't Throw Throwable":                             Exceptions/CantThrow
        "Cant Inherit Abstract Method":                      Classes/
↪CantInheritAbstractMethod
        "Cant Instantiate Class":                            Classes/
↪CantInstantiateClass
        "Case Insensitive Constants":                        Constants/
↪CaseInsensitiveConstants
        "Cast To Boolean":                                   Structures/CastToBoolean
        "Casting Ternary":                                   Structures/
↪CastingTernary
        "Catch Overwrite Variable":                          Structures/
↪CatchShadowsVariable
        "Check All Types":                                   Structures/CheckAllTypes
        "Check JSON":                                        Structures/CheckJson
        "Check On __Call Usage":                             Classes/CheckOnCallUsage
        "Child Class Removes Typehint":                      Classes/
↪ChildRemoveTypehint
        "Class Function Confusion":                          Php/
↪ClassFunctionConfusion
        "Class Should Be Final By Ocramius":                 Classes/FinalByOcramius
        "Class, Interface Or Trait With Identical Names":    Classes/CitSameName
        "Classes Mutually Extending Each Other":             Classes/MutualExtension
        "Clone With Non-Object":                             Classes/
↪CloneWithNonObject
        "Common Alternatives":                               Structures/
↪CommonAlternatives
        "Compact Inexistant Variable":                       Php/CompactInexistant
        "Compare Hash":                                      Security/CompareHash
        "Compared Comparison":                               Structures/
↪ComparedComparison
        "Concat And Addition":                               Php/ConcatAndAddition
        "Concat Empty String":                               Structures/ConcatEmpty
        "Concrete Visibility":                               Interfaces/
↪ConcreteVisibility
```

(continues on next page)

(continued from previous page)

```
        "Configure Extract":                                  Security/
↪ConfigureExtract
        "Const Visibility Usage":                             Classes/
↪ConstVisibilityUsage
        "Constants Created Outside Its Namespace":            Constants/
↪CreatedOutsideItsNamespace
        "Constants With Strange Names":                       Constants/
↪ConstantStrangeNames
        "Continue Is For Loop":                               Structures/
↪ContinueIsForLoop
        "Could Be Else":                                      Structures/CouldBeElse
        "Could Be Static":                                    Structures/CouldBeStatic
        "Could Use Short Assignation":                        Structures/
↪CouldUseShortAssignation
        "Could Use __DIR__":                                  Structures/CouldUseDir
        "Could Use self":                                     Classes/ShouldUseSelf
        "Could Use str_repeat()":                             Structures/
↪CouldUseStrrepeat
        "Crc32() Might Be Negative":                          Php/Crc32MightBeNegative
        "Dangling Array References":                          Structures/
↪DanglingArrayReferences
        "Deep Definitions":                                   Functions/
↪DeepDefinitions
        "Define With Array":                                  Php/DefineWithArray
        "Deprecated Functions":                               Php/Deprecated
        "Direct Call To __clone()":                           Php/DirectCallToClone
        "Direct Injection":                                   Security/DirectInjection
        "Don't Change Incomings":                             Structures/
↪NoChangeIncomingVariables
        "Don't Echo Error":                                   Security/DontEchoError
        "Don't Read And Write In One Expression":             Structures/
↪DontReadAndWriteInOneExpression
        "Don't Send $this In Constructor":                    Classes/
↪DontSendThisInConstructor
        "Don't Unset Properties":                             Classes/
↪DontUnsetProperties
        "Dont Change The Blind Var":                          Structures/
↪DontChangeBlindKey
        "Dont Mix ++":                                        Structures/
↪DontMixPlusPlus
        "Double Assignation":                                 Structures/
↪DoubleAssignation
        "Dynamic Library Loading":                            Security/DynamicDl
        "Echo With Concat":                                   Structures/
↪EchoWithConcat
        "Else If Versus Elseif":                              Structures/ElseIfElseif
        "Empty Blocks":                                       Structures/EmptyBlocks
        "Empty Instructions":                                 Structures/EmptyLines
        "Empty Interfaces":                                   Interfaces/
↪EmptyInterface
        "Empty Namespace":                                    Namespaces/
↪EmptyNamespace
        "Empty Traits":                                       Traits/EmptyTrait
        "Empty Try Catch":                                    Structures/EmptyTryCatch
        "Encoded Simple Letters":                             Security/EncodedLetters
        "Eval() Usage":                                       Structures/EvalUsage
        "Exception Order":                                    Exceptions/AlreadyCaught
```

(continues on next page)

(continued from previous page)

```
        "Exit() Usage":                                     Structures/ExitUsage
        "Failed Substr Comparison":                         Structures/
↪FailingSubstrComparison
        "Flexible Heredoc":                                 Php/FlexibleHeredoc
        "Foreach On Object":                                Php/ForeachObject
        "Foreach Reference Is Not Modified":                Structures/
↪ForeachReferenceIsNotModified
        "Forgotten Visibility":                             Classes/NonPpp
        "Forgotten Whitespace":                             Structures/
↪ForgottenWhiteSpace
        "Fully Qualified Constants":                        Namespaces/
↪ConstantFullyQualified
        "Functions/BadTypehintRelay":                       Functions/
↪BadTypehintRelay
        "Global Usage":                                     Structures/GlobalUsage
        "Group Use Declaration":                            Php/GroupUseDeclaration
        "Group Use Trailing Comma":                         Php/
↪GroupUseTrailingComma
        "Hash Algorithms Incompatible With PHP 5.3":        Php/HashAlgos53
        "Hash Algorithms":                                  Php/HashAlgos
        "Hash Will Use Objects":                            Php/HashUsesObjects
        "Hexadecimal In String":                            Type/HexadecimalString
        "Hidden Use Expression":                            Namespaces/HiddenUse
        "Htmlentities Calls":                               Structures/
↪Htmlentitiescall
        "Identical Conditions":                             Structures/
↪IdenticalConditions
        "Identical On Both Sides":                          Structures/
↪IdenticalOnBothSides
        "If With Same Conditions":                          Structures/
↪IfWithSameConditions
        "Illegal Name For Method":                          Classes/WrongName
        "Implement Is For Interface":                       Classes/
↪ImplementIsForInterface
        "Implemented Methods Are Public":                   Classes/
↪ImplementedMethodsArePublic
        "Implicit Global":                                  Structures/
↪ImplicitGlobal
        "Implied If":                                       Structures/ImpliedIf
        "Inclusion Wrong Case":                             Files/InclusionWrongCase
        "Incompatible Signature Methods":                   Classes/
↪IncompatibleSignature
        "Incompilable Files":                               Php/Incompilable
        "Indirect Injection":                               Security/
↪IndirectInjection
        "Integer As Property":                              Classes/
↪IntegerAsProperty
        "Integer Conversion":                               Security/
↪IntegerConversion
        "Invalid Class Name":                               Classes/WrongCase
        "Invalid Constant Name":                            Constants/InvalidName
        "Invalid Pack Format":                              Structures/
↪InvalidPackFormat
        "Invalid Regex":                                    Structures/InvalidRegex
        "Is Actually Zero":                                 Structures/IsZero
        "List Short Syntax":                                Php/ListShortSyntax
        "List With Appends":                                Php/ListWithAppends
```

(continues on next page)

(continued from previous page)

```
        "List With Reference":                               Php/ListWithReference
        "Logical Mistakes":                                  Structures/
↪LogicalMistakes
        "Logical Should Use Symbolic Operators":             Php/LogicalInLetters
        "Lone Blocks":                                       Structures/LoneBlock
        "Lost References":                                   Variables/LostReferences
        "Make Global A Property":                            Classes/
↪MakeGlobalAProperty
        "Method Collision Traits":                           Traits/
↪MethodCollisionTraits
        "Method Signature Must Be Compatible":               Classes/
↪MethodSignatureMustBeCompatible
        "Minus One On Error":                                Security/MinusOneOnError
        "Mismatch Type And Default":                         Functions/
↪MismatchTypeAndDefault
        "Mismatched Default Arguments":                      Functions/
↪MismatchedDefaultArguments
        "Mismatched Ternary Alternatives":                   Structures/
↪MismatchedTernary
        "Mismatched Typehint":                               Functions/
↪MismatchedTypehint
        "Missing Cases In Switch":                           Structures/MissingCases
        "Missing Include":                                   Files/MissingInclude
        "Missing New ?":                                     Structures/MissingNew
        "Missing Parenthesis":                               Structures/
↪MissingParenthesis
        "Mixed Concat And Interpolation":                    Structures/
↪MixedConcatInterpolation
        "Mkdir Default":                                     Security/MkdirDefault
        "Multiple Alias Definitions Per File":               Namespaces/
↪MultipleAliasDefinitionPerFile
        "Multiple Class Declarations":                       Classes/
↪MultipleDeclarations
        "Multiple Constant Definition":                      Constants/
↪MultipleConstantDefinition
        "Multiple Exceptions Catch()":                       Exceptions/MultipleCatch
        "Multiple Identical Trait Or Interface":             Classes/
↪MultipleTraitOrInterface
        "Multiple Index Definition":                         Arrays/
↪MultipleIdenticalKeys
        "Multiple Type Variable":                            Structures/
↪MultipleTypeVariable
        "Multiples Identical Case":                          Structures/
↪MultipleDefinedCase
        "Multiply By One":                                   Structures/MultiplyByOne
        "Must Call Parent Constructor":                      Php/
↪MustCallParentConstructor
        "Must Return Methods":                               Functions/MustReturn
        "Negative Power":                                    Structures/NegativePow
        "Nested Ternary":                                    Structures/NestedTernary
        "Never Used Parameter":                              Functions/
↪NeverUsedParameter
        "New Constants In PHP 7.2":                          Php/Php72NewConstants
        "New Functions In PHP 7.0":                          Php/Php70NewFunctions
        "New Functions In PHP 7.1":                          Php/Php71NewFunctions
        "New Functions In PHP 7.2":                          Php/Php72NewFunctions
        "New Functions In PHP 7.3":                          Php/Php73NewFunctions
```

(continues on next page)

(continued from previous page)

```
        "Next Month Trap":                                   Structures/NextMonthTrap
        "No Choice":                                         Structures/NoChoice
        "No Direct Call To Magic Method":                    Classes/
↪DirectCallToMagicMethod
        "No Direct Usage":                                   Structures/NoDirectUsage
        "No Empty Regex":                                    Structures/NoEmptyRegex
        "No Hardcoded Hash":                                 Structures/
↪NoHardcodedHash
        "No Hardcoded Ip":                                   Structures/NoHardcodedIp
        "No Hardcoded Path":                                 Structures/
↪NoHardcodedPath
        "No Hardcoded Port":                                 Structures/
↪NoHardcodedPort
        "No Magic With Array":                               Classes/NoMagicWithArray
        "No Parenthesis For Language Construct":             Structures/
↪NoParenthesisForLanguageConstruct
        "No Real Comparison":                                Type/NoRealComparison
        "No Reference For Ternary":                          Php/
↪NoReferenceForTernary
        "No Reference On Left Side":                         Structures/
↪NoReferenceOnLeft
        "No Return For Generator":                           Php/NoReturnForGenerator
        "No Return Or Throw In Finally":                     Structures/
↪NoReturnInFinally
        "No Return Used":                                    Functions/NoReturnUsed
        "No Self Referencing Constant":                      Classes/
↪NoSelfReferencingConstant
        "No String With Append":                             Php/NoStringWithAppend
        "No Substr Minus One":                               Php/NoSubstrMinusOne
        "No Substr() One":                                   Structures/NoSubstrOne
        "No get_class() With Null":                          Structures/
↪NoGetClassNull
        "No isset() With empty()":                           Structures/
↪NoIssetWithEmpty
        "Non Ascii Variables":                               Variables/
↪VariableNonascii
        "Non Static Methods Called In A Static":             Classes/
↪NonStaticMethodsCalledStatic
        "Non-constant Index In Array":                       Arrays/NonConstantArray
        "Not A Scalar Type":                                 Php/NotScalarType
        "Not Not":                                           Structures/NotNot
        "Objects Don't Need References":                     Structures/
↪ObjectReferences
        "Old Style Constructor":                             Classes/
↪OldStyleConstructor
        "Old Style __autoload()":                            Php/oldAutoloadUsage
        "One Variable String":                               Type/OneVariableStrings
        "Only Variable For Reference":                       Functions/
↪OnlyVariableForReference
        "Only Variable Passed By Reference":                 Functions/
↪OnlyVariablePassedByReference
        "Only Variable Returned By Reference":               Structures/
↪OnlyVariableReturnedByReference
        "Or Die":                                            Structures/OrDie
        "Overwritten Exceptions":                            Exceptions/
↪OverwriteException
        "Overwritten Literals":                              Variables/
↪OverwrittenLiterals
```

(continues on next page)

(continued from previous page)

```
        "PHP 7.0 New Classes":                               Php/Php70NewClasses
        "PHP 7.0 New Interfaces":                            Php/Php70NewInterfaces
        "PHP 7.0 Removed Directives":                        Php/
↪Php70RemovedDirective
        "PHP 7.0 Removed Functions":                         Php/
↪Php70RemovedFunctions
        "PHP 7.0 Scalar Typehints":                          Php/PHP70scalartypehints
        "PHP 7.1 Microseconds":                              Php/Php71microseconds
        "PHP 7.1 Removed Directives":                        Php/
↪Php71RemovedDirective
        "PHP 7.1 Scalar Typehints":                          Php/PHP71scalartypehints
        "PHP 7.2 Deprecations":                              Php/Php72Deprecation
        "PHP 7.2 Object Keyword":                            Php/Php72ObjectKeyword
        "PHP 7.2 Removed Functions":                         Php/
↪Php72RemovedFunctions
        "PHP 7.2 Scalar Typehints":                          Php/PHP72scalartypehints
        "PHP 7.3 Last Empty Argument":                       Php/
↪PHP73LastEmptyArgument
        "PHP 7.3 Removed Functions":                         Php/
↪Php73RemovedFunctions
        "PHP7 Dirname":                                      Structures/PHP7Dirname
        "Parent First":                                      Classes/ParentFirst
        "Parent, Static Or Self Outside Class":              Classes/PssWithoutClass
        "Parenthesis As Parameter":                          Php/
↪ParenthesisAsParameter
        "Pathinfo() Returns May Vary":                       Php/PathinfoReturns
        "Php 7 Indirect Expression":                         Variables/
↪Php7IndirectExpression
        "Php 7.1 New Class":                                 Php/Php71NewClasses
        "Php 7.2 New Class":                                 Php/Php72NewClasses
        "Php7 Relaxed Keyword":                              Php/Php7RelaxedKeyword
        "Phpinfo":                                           Structures/PhpinfoUsage
        "Possible Infinite Loop":                            Structures/
↪PossibleInfiniteLoop
        "Possible Missing Subpattern":                       Php/MissingSubpattern
        "Preprocessable":                                    Structures/
↪ShouldPreprocess
        "Print And Die":                                     Structures/PrintAndDie
        "Printf Number Of Arguments":                        Structures/
↪PrintfArguments
        "Property Could Be Local":                           Classes/
↪PropertyCouldBeLocal
        "Queries In Loops":                                  Structures/QueriesInLoop
        "Random Without Try":                                Structures/
↪RandomWithoutTry
        "Redeclared PHP Functions":                          Functions/
↪RedeclaredPhpFunction
        "Redefined Class Constants":                         Classes/
↪RedefinedConstants
        "Redefined Default":                                 Classes/RedefinedDefault
        "Redefined Private Property":                        Classes/
↪RedefinedPrivateProperty
        "Register Globals":                                  Security/RegisterGlobals
        "Repeated Interface":                                Interfaces/
↪RepeatedInterface
        "Repeated Regex":                                    Structures/RepeatedRegex
        "Repeated print()":                                  Structures/RepeatedPrint
```

(continues on next page)

(continued from previous page)

```
        "Results May Be Missing":                            Structures/
↪ResultMayBeMissing
        "Rethrown Exceptions":                               Exceptions/Rethrown
        "Return True False":                                 Structures/
↪ReturnTrueFalse
        "Safe Curl Options":                                 Security/CurlOptions
        "Safe HTTP Headers":                                 Security/SafeHttpHeaders
        "Same Variables Foreach":                            Structures/
↪AutoUnsetForeach
        "Scalar Or Object Property":                         Classes/
↪ScalarOrObjectProperty
        "Self Using Trait":                                  Traits/SelfUsingTrait
        "Session Lazy Write":                                Security/
↪SessionLazyWrite
        "Set Cookie Safe Arguments":                         Security/SetCookieArgs
        "Setlocale() Uses Constants":                        Structures/
↪SetlocaleNeedsConstants
        "Several Instructions On The Same Line":             Structures/
↪OneLineTwoInstructions
        "Short Open Tags":                                   Php/ShortOpenTagRequired
        "Should Chain Exception":                            Structures/
↪ShouldChainException
        "Should Make Alias":                                 Namespaces/
↪ShouldMakeAlias
        "Should Typecast":                                   Type/ShouldTypecast
        "Should Use Constants":                              Functions/
↪ShouldUseConstants
        "Should Use Prepared Statement":                     Security/
↪ShouldUsePreparedStatement
        "Should Use SetCookie()":                            Php/UseSetCookie
        "Should Yield With Key":                             Functions/
↪ShouldYieldWithKey
        "Silently Cast Integer":                             Type/SilentlyCastInteger
        "Sqlite3 Requires Single Quotes":                    Security/
↪Sqlite3RequiresSingleQuotes
        "Static Methods Can't Contain $this":                Classes/
↪StaticContainsThis
        "Strange Name For Constants":                        Constants/StrangeName
        "Strange Name For Variables":                        Variables/StrangeName
        "String Initialization":                             Arrays/
↪StringInitialization
        "String May Hold A Variable":                        Type/StringHoldAVariable
        "Strings With Strange Space":                        Type/
↪StringWithStrangeSpace
        "Strpos()-like Comparison":                          Structures/StrposCompare
        "Strtr Arguments":                                   Php/StrtrArguments
        "Suspicious Comparison":                             Structures/
↪SuspiciousComparison
        "Switch Fallthrough":                                Structures/Fallthrough
        "Switch To Switch":                                  Structures/
↪SwitchToSwitch
        "Switch Without Default":                            Structures/
↪SwitchWithoutDefault
        "Ternary In Concat":                                 Structures/
↪TernaryInConcat
        "Test Then Cast":                                    Structures/TestThenCast
        "Throw Functioncall":                                Exceptions/
↪ThrowFunctioncall
```

(continues on next page)

(continued from previous page)

```
        "Throw In Destruct":                                Classes/ThrowInDestruct
        "Throws An Assignement":                            Structures/
↪ThrowsAndAssign
        "Timestamp Difference":                             Structures/
↪TimestampDifference
        "Too Many Finds":                                   Classes/TooManyFinds
        "Too Many Native Calls":                            Php/TooManyNativeCalls
        "Trailing Comma In Calls":                          Php/TrailingComma
        "Traits/TraitNotFound":                             Traits/TraitNotFound
        "Typehint Must Be Returned":                        Functions/
↪TypehintMustBeReturned
        "Typehinted References":                            Functions/
↪TypehintedReferences
        "Unchecked Resources":                              Structures/
↪UncheckedResources
        "Unconditional Break In Loop":                      Structures/
↪UnconditionLoopBreak
        "Undeclared Static Property":                       Classes/
↪UndeclaredStaticProperty
        "Undefined Constants":                              Constants/
↪UndefinedConstants
        "Undefined Insteadof":                              Traits/
↪UndefinedInsteadof
        "Undefined static:: Or self::":                     Classes/
↪UndefinedStaticMP
        "Unicode Escape Syntax":                            Php/UnicodeEscapeSyntax
        "Unknown Pcre2 Option":                             Php/UnknownPcre2Option
        "Unkown Regex Options":                             Structures/
↪UnknownPregOption
        "Unpreprocessed Values":                            Structures/
↪Unpreprocessed
        "Unreachable Code":                                 Structures/
↪UnreachableCode
        "Unset In Foreach":                                 Structures/
↪UnsetInForeach
        "Unthrown Exception":                               Exceptions/Unthrown
        "Unused Constants":                                 Constants/
↪UnusedConstants
        "Unused Global":                                    Structures/UnusedGlobal
        "Unused Inherited Variable In Closure":             Functions/
↪UnusedInheritedVariable
        "Unused Interfaces":                                Interfaces/
↪UnusedInterfaces
        "Unused Label":                                     Structures/UnusedLabel
        "Unused Private Methods":                           Classes/
↪UnusedPrivateMethod
        "Unused Private Properties":                        Classes/
↪UnusedPrivateProperty
        "Unused Returned Value":                            Functions/
↪UnusedReturnedValue
        "Upload Filename Injection":                        Security/
↪UploadFilenameInjection
        "Use Constant As Arguments":                        Functions/
↪UseConstantAsArguments
        "Use Constant":                                     Structures/UseConstant
        "Use Instanceof":                                   Classes/UseInstanceof
        "Use Nullable Type":                                Php/UseNullableType
```

(continues on next page)

(continued from previous page)

```
        "Use PHP Object API":                                Php/UseObjectApi
        "Use Pathinfo":                                      Php/UsePathinfo
        "Use System Tmp":                                    Structures/UseSystemTmp
        "Use With Fully Qualified Name":                     Namespaces/
↪UseWithFullyQualifiedNS
        "Use const":                                         Constants/
↪ConstRecommended
        "Use random_int()":                                  Php/BetterRand
        "Used Once Variables":                               Variables/
↪VariableUsedOnce
        "Useless Abstract Class":                            Classes/UselessAbstract
        "Useless Alias":                                     Traits/UselessAlias
        "Useless Brackets":                                  Structures/
↪UselessBrackets
        "Useless Casting":                                   Structures/
↪UselessCasting
        "Useless Constructor":                               Classes/
↪UselessConstructor
        "Useless Final":                                     Classes/UselessFinal
        "Useless Global":                                    Structures/UselessGlobal
        "Useless Instructions":                              Structures/
↪UselessInstruction
        "Useless Interfaces":                                Interfaces/
↪UselessInterfaces
        "Useless Parenthesis":                               Structures/
↪UselessParenthesis
        "Useless Return":                                    Functions/UselessReturn
        "Useless Switch":                                    Structures/UselessSwitch
        "Useless Unset":                                     Structures/UselessUnset
        "Var Keyword":                                       Classes/OldStyleVar
        "Weak Typing":                                       Classes/WeakType
        "While(List() = Each())":                            Structures/WhileListEach
        "Wrong Number Of Arguments":                         Functions/
↪WrongNumberOfArguments
        "Wrong Optional Parameter":                          Functions/
↪WrongOptionalParameter
        "Wrong Parameter Type":                              Php/
↪InternalParameterType
        "Wrong Range Check":                                 Structures/WrongRange
        "Wrong fopen() Mode":                                Php/FopenMode
        "__DIR__ Then Slash":                                Structures/DirThenSlash
        "__toString() Throws Exception":                     Structures/
↪toStringThrowsException
        "error_reporting() With Integers":                   Structures/
↪ErrorReportingWithInteger
        "eval() Without Try":                                Structures/
↪EvalWithoutTry
        "ext/ereg":                                          Extensions/Extereg
        "ext/mcrypt":                                        Extensions/Extmcrypt
        "filter_input() As A Source":                        Security/
↪FilterInputSource
        "func_get_arg() Modified":                           Functions/
↪funcGetArgModified
        "include_once() Usage":                              Structures/OnceUsage
        "isset() With Constant":                             Structures/
↪IssetWithConstant
        "list() May Omit Variables":                         Structures/ListOmissions
```

(continues on next page)

(continued from previous page)

```
        "move_uploaded_file Instead Of copy":                Security/
↪MoveUploadedFile
        "parse_str() Warning":                               Security/
↪parseUrlWithoutParameters
        "preg_replace With Option e":                        Structures/pregOptionE
        "self, parent, static Outside Class":                Classes/
↪NoPSSOutsideClass
        "set_exception_handler() Warning":                   Php/
↪SetExceptionHandlerPHP7
        "var_dump()... Usage":                               Structures/VardumpUsage
    ruleset_1: # 1 errors found
        "Constant Class":                                    Classes/ConstantClass
        "Could Be Abstract Class":                           Classes/
↪CouldBeAbstractClass
        "Dependant Trait":                                   Traits/DependantTrait
        "Double Instructions":                               Structures/
↪DoubleInstruction
        "Drop Else After Return":                            Structures/
↪DropElseAfterReturn
        "Empty Classes":                                     Classes/EmptyClass
        "Forgotten Thrown":                                  Exceptions/
↪ForgottenThrown
        "Inconsistent Elseif":                               Structures/
↪InconsistentElseif
        "Instantiating Abstract Class":                      Classes/
↪InstantiatingAbstractClass
        "List With Keys":                                    Php/ListWithKeys
        "Logical To in_array":                               Performances/
↪LogicalToInArray
        "No Need For Else":                                  Structures/NoNeedForElse
        "Same Conditions In Condition":                      Structures/
↪SameConditions
        "Should Use session_regenerateid()":                 Security/
↪ShouldUseSessionRegenerateId
        "Static Loop":                                       Structures/StaticLoop
        "Too Many Injections":                               Classes/
↪TooManyInjections
        "Undefined Caught Exceptions":                       Exceptions/
↪CaughtButNotThrown
        "Unresolved Catch":                                  Classes/UnresolvedCatch
        "Unserialize Second Arg":                            Security/
↪UnserializeSecondArg
        "Use Positive Condition":                            Structures/
↪UsePositiveCondition
        "Useless Catch":                                     Exceptions/UselessCatch
        "Useless Check":                                     Structures/UselessCheck
    ruleset_2: # 2 errors found
        "Always Anchor Regex":                               Security/AnchorRegex
        "Forgotten Interface":                               Interfaces/
↪CouldUseInterface
        "No Class As Typehint":                              Functions/
↪NoClassAsTypehint
        "No array_merge() In Loops":                         Performances/
↪ArrayMergeInLoops
        "Pre-increment":                                     Performances/
↪PrePostIncrement
        "Randomly Sorted Arrays":                            Arrays/
↪RandomlySortedLiterals
```

(continues on next page)

(continued from previous page)

```
        "Should Make Ternary":                               Structures/
↪ShouldMakeTernary
        "Should Use Coalesce":                               Php/ShouldUseCoalesce
        "Use === null":                                      Php/IsnullVsEqualNull
   ruleset_3: # 3 errors found
        "@ Operator":                                        Structures/Noscream
        "Indices Are Int Or String":                         Structures/
↪IndicesAreIntOrString
        "Modernize Empty With Expression":                   Structures/ModernEmpty
        "Property Variable Confusion":                       Structures/
↪PropertyVariableConfusion
        "Too Many Local Variables":                          Functions/
↪TooManyLocalVariables
        "Unused Classes":                                    Classes/UnusedClass
        "Usort Sorting In PHP 7.0":                          Php/UsortSorting
   ruleset_4: # 4 errors found
        "Buried Assignation":                                Structures/
↪BuriedAssignation
        "Identical Consecutive Expression":                  Structures/
↪IdenticalConsecutive
        "Nested Ifthen":                                     Structures/NestedIfthen
        "No Boolean As Default":                             Functions/
↪NoBooleanAsDefault
        "Use Named Boolean In Argument Definition":          Functions/
↪AvoidBooleanArgument
   ruleset_5: # 5 errors found
        "Avoid Optional Properties":                         Classes/
↪AvoidOptionalProperties
        "Empty Function":                                    Functions/EmptyFunction
        "Relay Function":                                    Functions/RelayFunction
        "Strict Comparison With Booleans":                   Structures/
↪BooleanStrictComparison
        "Use Class Operator":                                Classes/UseClassOperator
        "strpos() Too Much":                                 Performances/
↪StrposTooMuch
   ruleset_6: # 6 errors found
        "Used Once Property":                                Classes/UsedOnceProperty
   ruleset_7: # 7 errors found
        "No Class In Global":                                Php/NoClassInGlobal
        "Uncaught Exceptions":                               Exceptions/
↪UncaughtExceptions
        "Unused Functions":                                  Functions/
↪UnusedFunctions
        "Wrong Number Of Arguments In Methods":              Functions/
↪WrongNumberOfArgumentsMethods
   ruleset_8: # 8 errors found
        "Could Make A Function":                             Functions/
↪CouldCentralize
        "Insufficient Typehint":                             Functions/
↪InsufficientTypehint
        "Long Arguments":                                    Structures/LongArguments
        "Property Used In One Method Only":                  Classes/
↪PropertyUsedInOneMethodOnly
        "Static Methods Called From Object":                 Classes/
↪StaticMethodsCalledFromObject
   ruleset_9: # 9 errors found
        "PHP Keywords As Names":                             Php/ReservedNames
```

(continues on next page)

(continued from previous page)

```
        "Undefined Trait":                                   Traits/UndefinedTrait
        "Written Only Variables":                            Variables/
↪WrittenOnlyVariable
   ruleset_10: # 10 errors found
        "Bail Out Early":                                    Structures/BailOutEarly
        "Hardcoded Passwords":                               Functions/
↪HardcodedPasswords
        "Multiple Alias Definitions":                        Namespaces/
↪MultipleAliasDefinitions
   ruleset_11: # 11 errors found
        "Variable Is Not A Condition":                       Structures/
↪NoVariableIsACondition
   ruleset_13: # 13 errors found
        "Undefined Functions":                               Functions/
↪UndefinedFunctions
        "Unused Use":                                        Namespaces/UnusedUse
   ruleset_14: # 14 errors found
        "Iffectations":                                      Structures/Iffectation
        "No Public Access":                                  Classes/NoPublicAccess
   ruleset_16: # 16 errors found
        "Overwriting Variable":                              Variables/Overwriting
   ruleset_17: # 17 errors found
        "No Net For Xml Load":                               Security/NoNetForXmlLoad
        "Unresolved Instanceof":                             Classes/
↪UnresolvedInstanceof
   ruleset_21: # 21 errors found
        "Undefined Class Constants":                         Classes/
↪UndefinedConstants
   ruleset_27: # 27 errors found
        "Locally Unused Property":                           Classes/
↪LocallyUnusedProperty
        "Never Used Properties":                             Classes/
↪PropertyNeverUsed
   ruleset_35: # 35 errors found
        "Useless Referenced Argument":                       Functions/
↪UselessReferenceArgument
   ruleset_38: # 38 errors found
        "Uses Default Values":                               Functions/
↪UsesDefaultArguments
   ruleset_47: # 47 errors found
        "Unused Arguments":                                  Functions/
↪UnusedArguments
   ruleset_49: # 49 errors found
        "Undefined Properties":                              Classes/
↪UndefinedProperty
   ruleset_77: # 77 errors found
        "Undefined Parent":                                  Classes/
↪UndefinedParentMP
   ruleset_78: # 78 errors found
        "Undefined ::class":                                 Classes/
↪UndefinedStaticclass
   ruleset_82: # 82 errors found
        "Class Could Be Final":                              Classes/CouldBeFinal
   ruleset_86: # 86 errors found
        "Unused Protected Methods":                          Classes/
↪UnusedProtectedMethods
   ruleset_89: # 89 errors found
```

(continues on next page)

(continued from previous page)

```
        "Unresolved Classes":                                    Classes/
↪UnresolvedClasses
    ruleset_94: # 94 errors found
        "Used Once Variables (In Scope)":                        Variables/
↪VariableUsedOnceByContext
    ruleset_122: # 122 errors found
        "Method Could Be Static":                                Classes/CouldBeStatic
    ruleset_133: # 133 errors found
        "Should Use Local Class":                                Classes/ShouldUseThis
    ruleset_159: # 159 errors found
        "Undefined Interfaces":                                  Interfaces/
↪UndefinedInterfaces
    ruleset_160: # 160 errors found
        "Unused Methods":                                        Classes/UnusedMethods
    ruleset_183: # 183 errors found
        "Undefined Variable":                                    Variables/
↪UndefinedVariable
    ruleset_337: # 337 errors found
        "Unresolved Use":                                        Namespaces/UnresolvedUse
    ruleset_595: # 595 errors found
        "Undefined Classes":                                     Classes/UndefinedClasses
```

Exakatyaml is a Yaml report format.

Exakatyaml doesn't depend on themes.

11.4.12 File dependendies

This reports displays the file dependencies, based on definition usages.

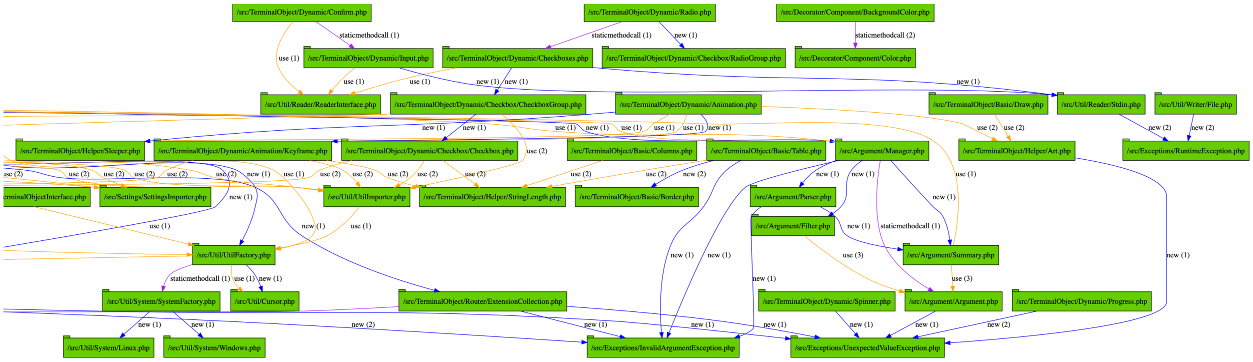
This report displays all dependencies between files. A file depends on another when it makes usage of one of its definitions : constant, functions, classes, traits, interfaces.

For example, *A.php* depends on *B.php*, because *A.php* uses the function *foo*, which is defined in the *B.php* file. On the other hand, *B.php* doesn't depends on *A.php*, as a function may be defined, but not used.

This diagram shows which files may be used without others.

The resulting diagram is a DOT file, which is readable with [Graphviz](https://www.graphviz.org/about/). Those viewers will display the diagram, and also convert it to other format, such as PNG, JPEG, PDF or others.

Another version of the same diagram is called Filedependencieshtml


/src/TerminalObject/Dynamic/Confirm.php
/src/TerminalObject/Dynamic/Radio.php
/src/Decorator/Component/BackgroundColor.php
/src/TerminalObject/Dynamic/Input.php
/src/TerminalObject/Dynamic/Checkboxes.php
/src/TerminalObject/Dynamic/Checkbox/RadioGroup.php
/src/Decorator/Component/Color.php
/src/Util/Reader/ReaderInterface.php
/src/TerminalObject/Dynamic/Checkbox/CheckboxGroup.php
/src/TerminalObject/Dynamic/Animation.php
/src/TerminalObject/Basic/Draw.php
/src/Util/Reader/Stdin.php
/src/Util/Writer/File.php
/src/TerminalObject/Helper/Sleeper.php
/src/TerminalObject/Dynamic/Animation/Keyframe.php
/src/TerminalObject/Dynamic/Checkbox/Checkbox.php
/src/TerminalObject/Basic/Columns.php
/src/TerminalObject/Basic/Table.php
/src/Argument/Manager.php
/src/TerminalObject/Helper/Art.php
/src/Exceptions/RuntimeException.php
erminalObjectInterface.php
/src/Settings/SettingsImporter.php
/src/Util/UtilImporter.php
/src/TerminalObject/Helper/StringLength.php
/src/TerminalObject/Basic/Border.php
/src/Argument/Parser.php
/src/Argument/Filter.php
/src/Util/UtilFactory.php
/src/Argument/Summary.php
/src/Util/System/SystemFactory.php
/src/Util/Cursor.php
/src/TerminalObject/Router/ExtensionCollection.php
/src/TerminalObject/Dynamic/Spinner.php
/src/Argument/Argument.php
/src/TerminalObject/Dynamic/Progress.php
/src/Util/System/Linux.php
/src/Util/System/Windows.php
/src/Exceptions/InvalidArgumentException.php
/src/Exceptions/UnexpectedValueException.php
staticmethodcall (1)
staticmethodcall (2)
use (1)
use (2)
use (3)
new (1)
new (2)

File dependendies is a DOT report format.

File dependendies doesn't depend on themes.

11.4.13 File dependendies HTML

This reports displays the file dependencies, based on definition usages.

This report displays all dependencies between files. A file depends on another when it makes usage of one of its definitions : constant, functions, classes, traits, interfaces.

For example, *A.php* depends on *B.php*, because *A.php* uses the function *foo*, which is defined in the *B.php* file. On the other hand, *B.php* doesn't depends on *A.php*, as a function may be defined, but not used.

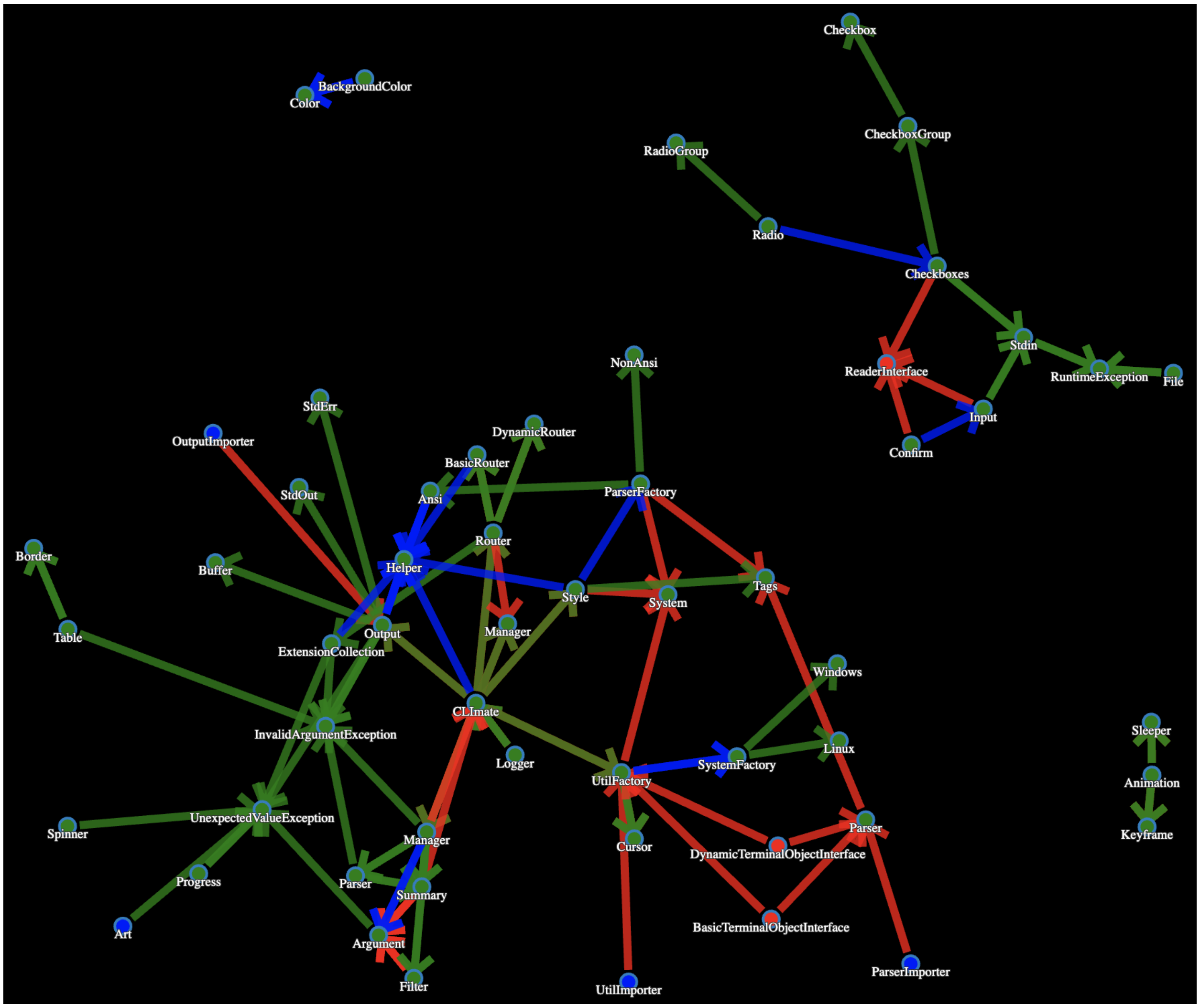
This diagram shows which files may be used without others.

The resulting diagram is in HTML file, which is readable with most browsers, from a web server.

Warning : for browser security reasons, the report will NOT load as a local file. It needs to be served by an HTTP server, so all resources are correctly located.

Warning : large applications (> 1000 files) will require a lot of resources to open.

Another version of the same diagram is called Filedependencies, and produces a DOT file



File dependendies HTML is a HTML report format.

File dependendies HTML doesn't depend on themes.

11.4.14 History

The History report collects meta information between audits. It saves the values from the current audit into a separate 'history.sqlite' database.

The history tables are the same as the dump.sqlite tables, except for the extra 'serial' table. Each audit comes with 3 identifiers :

- 'dump_timestamp' : this is a timmestamp taken when the dump was build
- 'dump_serial' : this is a serial number, based on the previous audit, and incremented by one. This is handy to keep the values in sequence
- 'dump_id' : this is a unique random id, which helps distinguish audits which may have inconsistency between serial or timestamp.

This report provides a 'history.sqlite' database. The following tables are inventoried :

- hash
- resultsCounts

History is a Sqlite report format.

History doesn't depend on themes.

11.4.15 Inventories

The Inventories report collects literals and names from the code.

This report provides the value, the file and line where a type of value is present.

The following values and names are inventoried :

- Variables
- Incoming Variables
- Session Variables
- Global Variables
- Date formats
- Constants
- Functions
- Classes
- Interface names
- Trait names
- Namespaces
- Exceptions
- Regex
- SQL queries
- URL
- Unicode blocks
- Integers

- Reals numbers
- Literal Arrays
- Strings

Every type of values is exported to a file. If no value of such type was found during the audit, the file only contains the headers. It is always produced.

```
Name,File,Line
0,/features/bootstrap/FeatureContext.php,61
10000,/features/bootstrap/FeatureContext.php,61
777,/features/bootstrap/FeatureContext.php,63
20,/features/bootstrap/FeatureContext.php,73
0,/features/bootstrap/FeatureContext.php,334
0,/features/bootstrap/FeatureContext.php,339
0,/features/bootstrap/FeatureContext.php,344
0,/features/bootstrap/FeatureContext.php,362
0,/features/bootstrap/FeatureContext.php,366
0,/features/bootstrap/FeatureContext.php,368
0,/features/bootstrap/FeatureContext.php,372
777,/features/bootstrap/FeatureContext.php,423
777,/features/bootstrap/FeatureContext.php,431
0,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,68
1,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,69
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,84
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,150
```

Inventories is a CSV report format.

Inventories depends on the following theme : Inventories.

11.4.16 Json

The JSON report exports in JSON format.

Simple Json format. It is a structured array with all results, described as object.

```
Filename => [
                errors   => count,
                warning  => count,
                fixable  => count,
                filename => string,
                message  => [
                    line => [
                        type,
                        source,
                        severity,
                        fixable,
                        message
                    ]
                ]
            ]
```

```
{
   "\/src\/Path\/To\/File.php":{
      "errors":0,
      "warnings":105,
```

(continues on next page)

(continued from previous page)

```
      "fixable":0,
      "filename":"\/src\/Path\/To\/File.php",
      "messages":{
         "55":[
            [
               {
                  "type":"warning",
                  "source":"Php/EllipsisUsage",
                  "severity":"Major",
                  "fixable":"fixable",
                  "message":"... Usage"
               }
            ]
         ],
         }
    }
}
```

Json is a Json report format.

Json accepts any arbitrary list of results.

11.4.17 Marmelab

The Marmelab report format data to use with a graphQL server.

Marmelab is a report format to build GraphQL server with exakat's results. Export the results of the audit in this JSON file, then use the [json-graphql-server](https://github.com/marmelab/json-graphql-server) to have a GraphQL server with all the results.

You may also learn more about GraphQL at [Introducing Json GraphQL Server](https://marmelab.com/blog/2017/07/12/json-graphql-server.html).

```
php exakat.phar report -p -format Marmelab -file marmelab
cp projects/myproject/marmelab.json path/to/marmelab
json-graphql-server db.json
```

Marmelab is a JSON report format.

Marmelab depends on the following theme : Analyze.

11.4.18 Meters

The Meters report export various dimensions of the audited code.

Exakat measures a large number of code dimensions, such as number of files, lines of code, tokens. All those are collected in this report.

```
{
```

loc: 95950, locTotal: 140260, files: 1824, tokens: 677213

}

Meters is a JSON report format.

Meters depends on the following theme : None.

11.4.19 Migration74

The Migration74 is the report dedicated to migrating PHP code to version 7.4.

The Migration74 report runs the backward incompatibilities tests for PHP 7.4, from a PHP 7.3 compatible code.

```
Name,File,Line
0,/features/bootstrap/FeatureContext.php,61
10000,/features/bootstrap/FeatureContext.php,61
777,/features/bootstrap/FeatureContext.php,63
20,/features/bootstrap/FeatureContext.php,73
0,/features/bootstrap/FeatureContext.php,334
0,/features/bootstrap/FeatureContext.php,339
0,/features/bootstrap/FeatureContext.php,344
0,/features/bootstrap/FeatureContext.php,362
0,/features/bootstrap/FeatureContext.php,366
0,/features/bootstrap/FeatureContext.php,368
0,/features/bootstrap/FeatureContext.php,372
777,/features/bootstrap/FeatureContext.php,423
777,/features/bootstrap/FeatureContext.php,431
0,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,68
1,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,69
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,84
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,150
```

Migration74 is a HTML report format.

Migration74 depends on the following 2 themes : CompatibilityPHP73, Suggestions.

11.4.20 Migration80

The Migration80 is the report dedicated to migrating PHP code to version 8.0.

The Migration 80 report runs the backward incompatibilities tests for PHP 8.0, from a PHP 7.4 compatible code.

```
Name,File,Line
0,/features/bootstrap/FeatureContext.php,61
10000,/features/bootstrap/FeatureContext.php,61
777,/features/bootstrap/FeatureContext.php,63
20,/features/bootstrap/FeatureContext.php,73
0,/features/bootstrap/FeatureContext.php,334
0,/features/bootstrap/FeatureContext.php,339
0,/features/bootstrap/FeatureContext.php,344
0,/features/bootstrap/FeatureContext.php,362
0,/features/bootstrap/FeatureContext.php,366
0,/features/bootstrap/FeatureContext.php,368
0,/features/bootstrap/FeatureContext.php,372
777,/features/bootstrap/FeatureContext.php,423
777,/features/bootstrap/FeatureContext.php,431
0,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,68
1,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,69
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,84
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,150
```

Migration80 is a HTML report format.

Migration80 depends on the following 2 themes : CompatibilityPHP80, Suggestions.

11.4.21 None

None is the empty report. It runs the report generating stack, but doesn't produce any result.

None is a utility report, aimed to test exakat's installation.

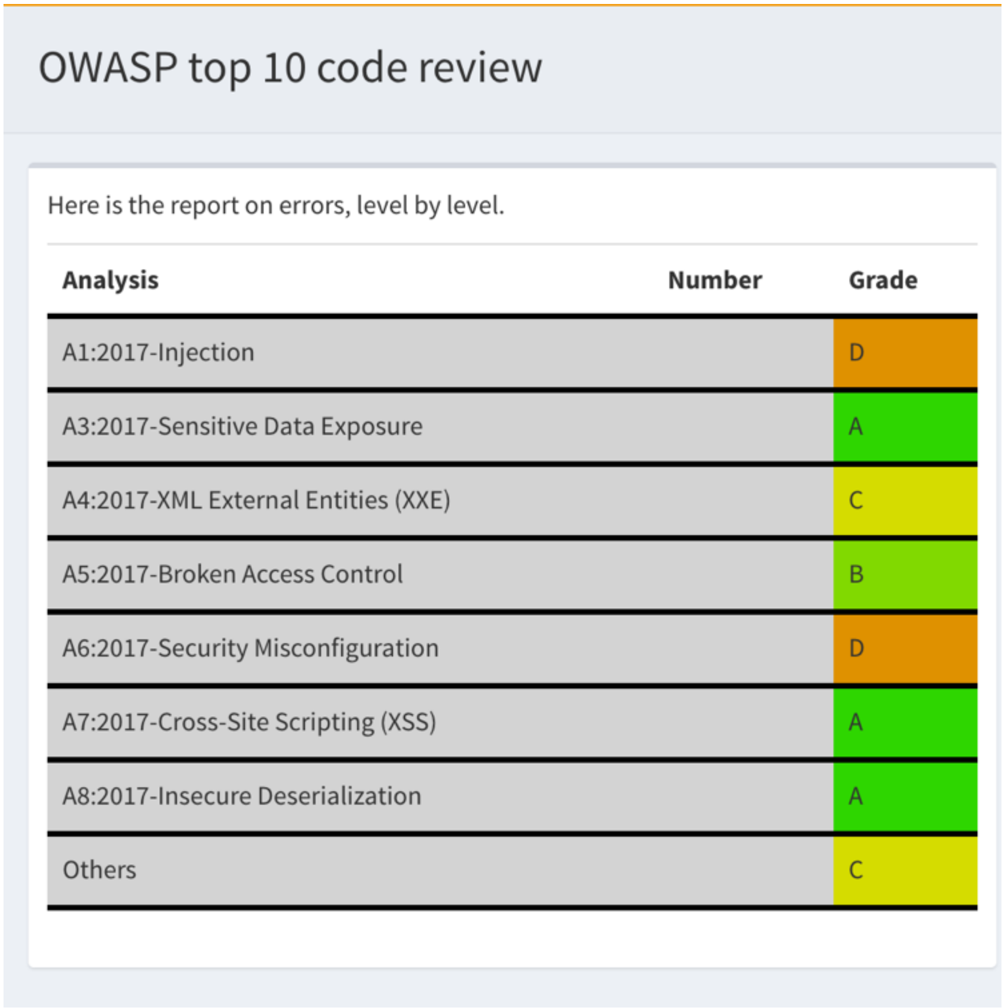
None is a None report format.

None depends on the following theme : Any.

11.4.22 Owasp

The OWASP report is a security report.

The OWASP report focuses on the [OWASP top 10](https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project). It reports all the security analysis, distributed across the 10 categories of vulnerabilities.



OWASP top 10 code review

Here is the report on errors, level by level.

Analysis	Number	Grade
A1:2017-Injection		D
A3:2017-Sensitive Data Exposure		A
A4:2017-XML External Entities (XXE)		C
A5:2017-Broken Access Control		B
A6:2017-Security Misconfiguration		D
A7:2017-Cross-Site Scripting (XSS)		A
A8:2017-Insecure Deserialization		A
Others		C

Owasp is a HTML report format.

Owasp depends on the following theme : Security.

11.4.23 Perfile

The Perfile report lays out the results file per file.

The Perfile report displays one result per line, grouped by file, and ordered by line number :

```
/path/from/project/root/to/file:line[space]name of analysis
```

This format is fast, and fitted for human review.

```
---------------------------------------------------------
 line  /themes/Rozier/Controllers/LoginController.php
---------------------------------------------------------
   34  Multiple Alias Definitions
   36  Unresolved Use
   43  Multiple Alias Definitions
   51  Class Could Be Final
   58  Undefined Interfaces
   81  Undefined Interfaces
   81  Unused Arguments
   81  Used Once Variables (In Scope)
   91  Undefined Interfaces
   91  Unused Arguments
   91  Used Once Variables (In Scope)
  101  Undefined Interfaces
  103  Nested Ifthen
  104  Unresolved Classes
  106  Buried Assignation
  106  Iffectations
  106  Use Positive Condition
  121  Uncaught Exceptions
  121  Unresolved Classes
  129  Uncaught Exceptions
---------------------------------------------------------
```

Perfile is a Text report format.

Perfile accepts any arbitrary list of results.

11.4.24 PhpCompilation

The PhpCompilation suggests a list of compilation directives when compiling the PHP binary, tailored for the code

PhpCompilation bases its selection on the code and its usage of features. PhpCompilation also recommends disabling unused standard extensions : this helps reducing the footprint of the binary, and prevents unused features to be available for intrusion. PhpCompilation is able to detects over 150 PHP extensions.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;
; Suggestion for php.ini ;
;;;;;;;;;;;;;;;;;;;;;;;;;;

; The directives below are selected based on the code provided.
; They only cover the related directives that may have an impact on the code
;
; The list may not be exhaustive
; The suggested values are not recommendations, and should be reviewed and adapted
;
```

(continues on next page)

(continued from previous page)

```
[date]
; It is not safe to rely on the system's timezone settings. Make sure the
; directive date.timezone is set in php.ini.
date.timezone = Europe/Amsterdam



[pcre]
; More information about pcre :
;http://php.net/manual/en/pcre.configuration.php



[standard]
; This sets the maximum amount of memory in bytes that a script is allowed to
; allocate. This helps prevent poorly written scripts for eating up all available
; memory on a server. It is recommended to set this as low as possible and avoid
; removing the limit.
memory_limit = 120

; This sets the maximum amount of time, in seconds, that a script is allowed to
; run. The lower the value, the better for the server, but also, the better has
; the script to be written. Avoid really large values that are only useful for
; admin, and set them per directory.
max_execution_time = 90

; Exposes to the world that PHP is installed on the server. For security reasons,
; it is better to keep this hidden.
expose_php = Off

; This determines whether errors should be printed to the screen as part of the
; output or if they should be hidden from the user.
display_errors = Off

; Set the error reporting level. Always set this high, so as to have the errors
; reported, and logged.
error_reporting = E_ALL

; Always log errors for future use
log_errors = On

; Name of the file where script errors should be logged.
error_log = Name of a writable file, suitable for logging.

; More information about standard :
;http://php.net/manual/en/info.configuration.php

; Name of the file where script errors should be logged.
disable_functions = curl_init,ftp_connect,ftp_ssl_connect,ldap_connect,mail,mysqli_
↪connect,mysqli_pconnect,pg_connect,pg_pconnect,socket_create,socket_accept,socket_
↪connect,socket_listen
disable_classes = mysqli
```

PhpCompilation is a Text report format.

PhpCompilation depends on the following theme : Appinfo.

11.4.25 PhpConfiguration

The PhpConfiguration suggests a list of directives to check when setting up the hosting server, tailored for the code

PhpConfiguration bases its selection on the code, and classic recommendations. For example, memory_limit or expose_php are always reported, though they have little impact in the code. Extensions also get a short list of important directive, and offer a link to the documentation for more documentation.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;
; Suggestion for php.ini ;
;;;;;;;;;;;;;;;;;;;;;;;;;;

; The directives below are selected based on the code provided.
; They only cover the related directives that may have an impact on the code
;
; The list may not be exhaustive
; The suggested values are not recommendations, and should be reviewed and adapted
;


[date]
; It is not safe to rely on the system's timezone settings. Make sure the
; directive date.timezone is set in php.ini.
date.timezone = Europe/Amsterdam



[pcre]
; More information about pcre :
;http://php.net/manual/en/pcre.configuration.php



[standard]
; This sets the maximum amount of memory in bytes that a script is allowed to
; allocate. This helps prevent poorly written scripts for eating up all available
; memory on a server. It is recommended to set this as low as possible and avoid
; removing the limit.
memory_limit = 120

; This sets the maximum amount of time, in seconds, that a script is allowed to
; run. The lower the value, the better for the server, but also, the better has
; the script to be written. Avoid really large values that are only useful for
; admin, and set them per directory.
max_execution_time = 90

; Exposes to the world that PHP is installed on the server. For security reasons,
; it is better to keep this hidden.
expose_php = Off

; This determines whether errors should be printed to the screen as part of the
; output or if they should be hidden from the user.
display_errors = Off

; Set the error reporting level. Always set this high, so as to have the errors
; reported, and logged.
error_reporting = E_ALL
```

(continues on next page)

(continued from previous page)

```
; Always log errors for future use
log_errors = On

; Name of the file where script errors should be logged.
error_log = Name of a writable file, suitable for logging.

; More information about standard :
;http://php.net/manual/en/info.configuration.php

; Name of the file where script errors should be logged.
disable_functions = curl_init,ftp_connect,ftp_ssl_connect,ldap_connect,mail,mysqli_
↪connect,mysqli_pconnect,pg_connect,pg_pconnect,socket_create,socket_accept,socket_
↪connect,socket_listen
disable_classes = mysqli
```

PhpConfiguration is a Text report format.

PhpConfiguration depends on the following theme : Appinfo.

11.4.26 Phpcity

The Phpcity report represents your code as a city.

Phpcity is a code visualisation tool : it displays the source code as a city, with districts and buildings. Ther will be high sky crappers, signaling large classes, entire districts of small blocks, large venues and isolated parks. Some imagination is welcome too.

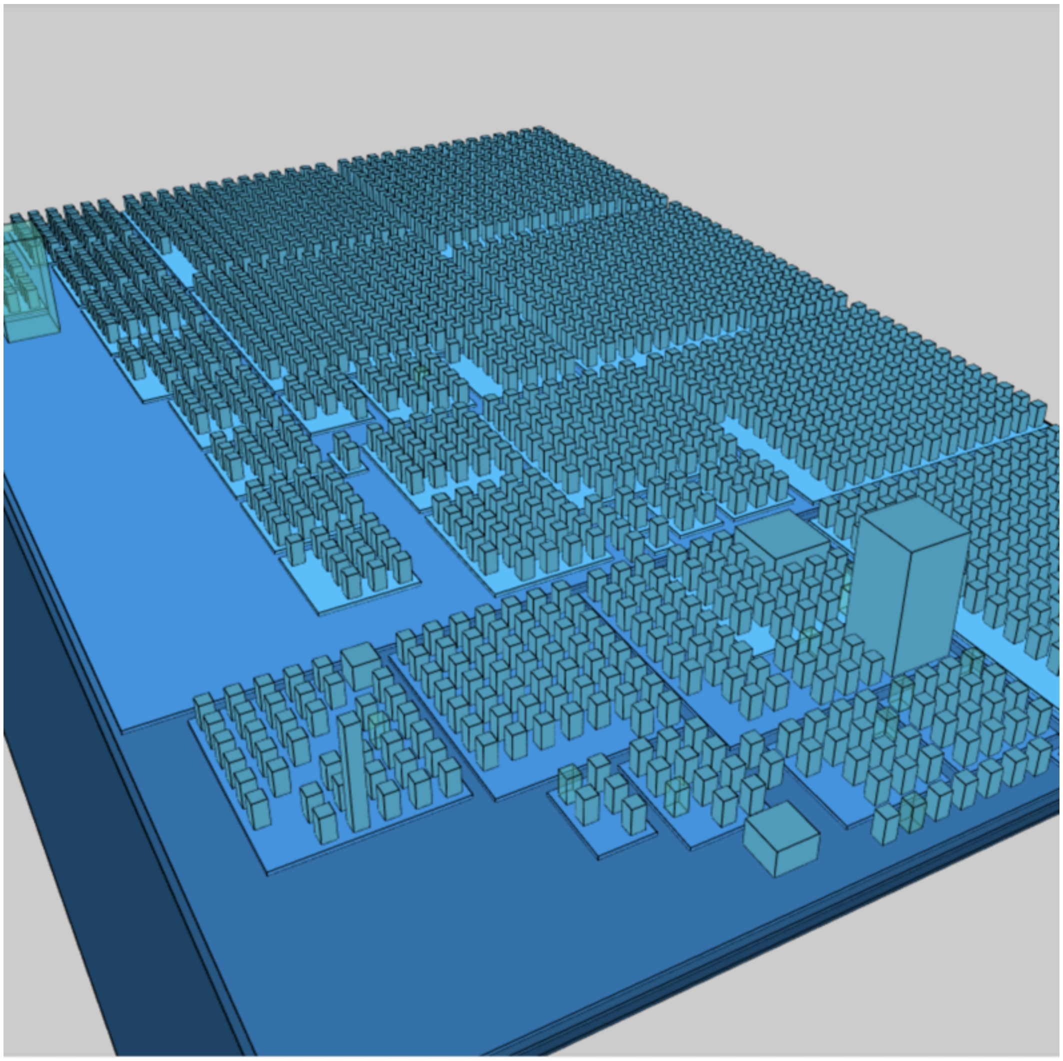
The original idea is Richard Wettel's [Code city](https://wettel.github.io/codecity.html), which has been adapted to many languages, including PHP. The PHP version is based on the open source [PHPcity project](https://github.com/adrianhuna/PHPCity), which is itself build with [JScity](https://github.com/ASERG-UFMG/JSCity/wiki/JSCITY).

To use this tool, run an exakat audit, then generate the 'PHPcity' report : *php exakat.phar report -p mycode -format PHPcity -v*

This generates the *exakat.phpcity.json* file, in the *projects/mycode/* folder.

You may test your own report online, at [Adrian Huna](https://github.com/adrianhuna)'s website, by [uploading the results](https://adrianhuna.github.io/PHPCity/) and seeing it live immediately.

Or, you can install the [PHPcity](https://github.com/adrianhuna/PHPCity) application, and load it locally.



Phpcity is a JSON report format.

Phpcity doesn't depend on themes.

11.4.27 Phpcsfixer

The Phpcsfixer report provides a configuration file for php-cs-fixer, that automatically fixes issues found in related analysis in exakat.

This report builds a configuration file for php-cs-fixer.

- *Use === null* : **is_null**
- *Else If Versus Elseif* : **elseif**
- *Multiple Unset()* : **combine_consecutive_unsets**
- Classes/DontUnsetProperties: **no_unset_on_property**
- *Use Constant* : **function_to_constant**
- *PHP7 Dirname* : **combine_nested_dirname**
- *Could Use __DIR__* : **dir_constant**

- *Isset Multiple Arguments* : **combine_consecutive_issets**
- *Logical Should Use Symbolic Operators* : **logical_operators**
- *Not Not* : **no_short_bool_cast**

PHP-cs-fixer is a tool to automatically fix PHP Coding Standards issues. Some of the modifications are more than purely coding standards, such has replacing `dirname(dirname($path))` with `dirname($path, 2)`.

Exakat builds a configuration file for php-cs-fixer, that will automatically fix a number of results from the audit. Here is the process :

- Run exakat audit
- Get Phpcsfixer report from exakat : `php exakat.phar report -p <project> -format Phpcsfixer`
- Update the target repository in the generated code
- Save this new configuration in a file called '.php_cs'
- Run php-cs-fixer on your code : `php php-cs-fixer.phar fix /path/to/code --dry-run`
- Fixed your code with php-cs-fixer : `php php-cs-fixer.phar fix /path/to/code`
- Run a new exakat audit

This configuration file should be reviewed before being used. In particular, the target files should be updated with the actual repository : this is the first part of the configuration.

It is also recommended to use the option '–dry-run' with php-cs-fixer to check the first run.

Php-cs-fixer runs fixes for coding standards : this reports focuses on potential fixes. It is recommended to complete this base report with extra coding conventions fixes. The building of a coding convention is outside the scope of this report.

Exakat may find different issues than php-cs-fixer : using this report reduces the number of reported issues, but may leave some issues unsolved. In that case, manual fixing is recommended.

Phpcsfixer is a JSON report format.

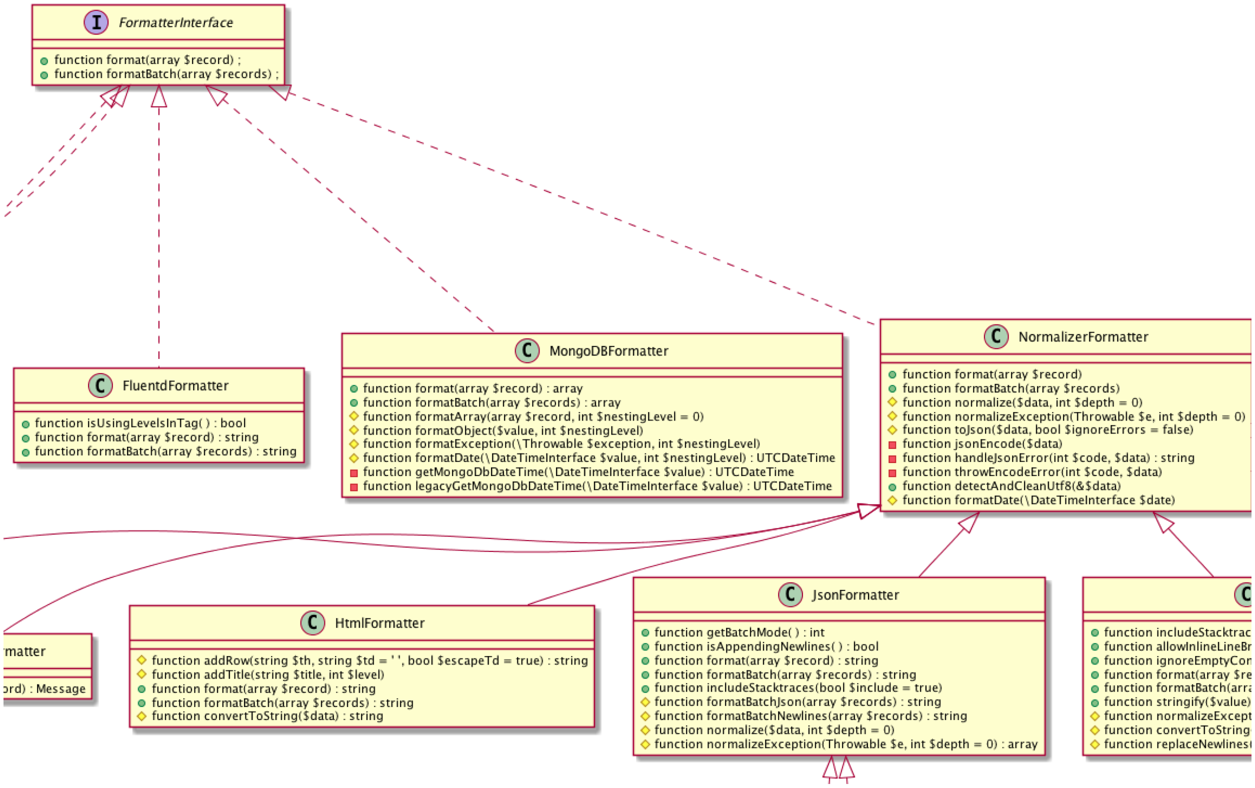
Phpcsfixer depends on the following theme : php-cs-fixable.

11.4.28 PlantUml

The PlantUml export data structure to PlantUml format.

This report produces a .puml file, compatible with [PlantUML](http://plantuml.com/).

PlantUML is an Open Source component that dislays class diagrams.


I FormatterInterface
function format(array $record) ;
function formatBatch(array $records) ;
C FluentdFormatter
function isUsingLevelsInTag( ) : bool
function format(array $record) : string
function formatBatch(array $records) : string
C MongoDBFormatter
function format(array $record) : array
function formatBatch(array $records) : array
function formatArray(array $record, int $nestingLevel = 0)
function formatObject($value, int $nestingLevel)
function formatException(\Throwable $exception, int $nestingLevel)
function formatDate(\DateTimeInterface $value, int $nestingLevel) : UTCDateTime
function getMongoDbDateTime(\DateTimeInterface $value) : UTCDateTime
function legacyGetMongoDbDateTime(\DateTimeInterface $value) : UTCDateTime
C NormalizerFormatter
function format(array $record)
function formatBatch(array $records)
function normalize($data, int $depth = 0)
function normalizeException(Throwable $e, int $depth = 0)
function toJson($data, bool $ignoreErrors = false)
function jsonEncode($data)
function handleJsonError(int $code, $data) : string
function throwEncodeError(int $code, $data)
function detectAndCleanUtf8(&$data)
function formatDate(\DateTimeInterface $date)
rmatter
ord) : Message
C HtmlFormatter
function addRow(string $th, string $td = ' ', bool $escapeTd = true) : string
function addTitle(string $title, int $level)
function format(array $record) : string
function formatBatch(array $records) : string
function convertToString($data) : string
C JsonFormatter
function getBatchMode( ) : int
function isAppendingNewlines( ) : bool
function format(array $record) : string
function formatBatch(array $records) : string
function includeStacktraces(bool $include = true)
function formatBatchJson(array $records) : string
function formatBatchNewlines(array $records) : string
function normalize($data, int $depth = 0)
function normalizeException(Throwable $e, int $depth = 0) : array
C
function includeStacktrac
function allowInlineLineBr
function ignoreEmptyCon
function format(array $re
function formatBatch(arra
function stringify($value)
function normalizeExcept
function convertToString
function replaceNewlines

PlantUml is a puml report format.

PlantUml doesn't depend on themes.

11.4.29 RadwellCode

The RadwellCode is a report based on Oliver Radwell's [PHP Do And Don't](https://blog.radwell.codes/2016/11/php-dos-donts-aka-programmers-dont-like/).

Note that all rules are not implemented, especially the 'coding conventions' ones, as this is beyond the scope of this tool.

```
/Phrozn/Vendor/Extra/scss.inc.php:594 Slow PHP built-in functions
/Phrozn/Vendor/Extra/scss.inc.php:2554 Too many nested if statements
/Phrozn/Vendor/Extra/scss.inc.php:1208 Long if-else blocks
/Phrozn/Vendor/Extra/scss.inc.php:1208 Too many nested if statements
/Phrozn/Vendor/Extra/scss.inc.php:3935 Wrong function / class name casing
/Phrozn/Vendor/Extra/scss.inc.php:3452 Too many nested if statements
/Phrozn/Site/View/OutputPath/Entry/Parametrized.php:58 Slow PHP built-in functions
/Phrozn/Runner/CommandLine/Callback/Init.php:82 Extra brackets and braces and quotes
```

RadwellCode is a Text report format.

RadwellCode depends on the following theme : RadwellCodes.

11.4.30 Rector

Suggest configuration for Rector refactoring tool.

The Rector report is a helper report for [Tomas Votruba](https://twitter.com/VotrubaT)'s [Rector](https://getrector.org/) tool.

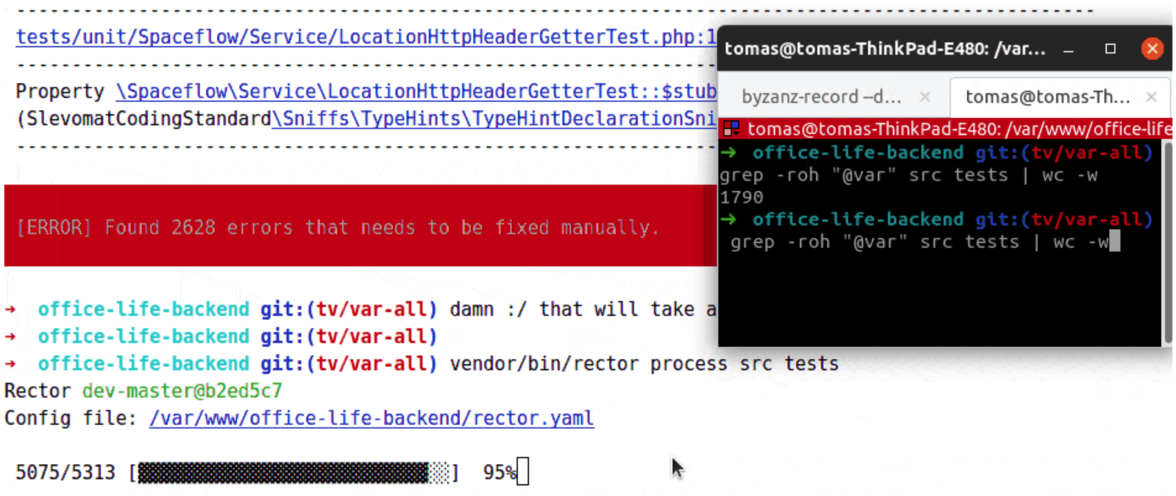
Some issues spotted by Exakat may be fixed automagically by Rector. Rector offers more than 550 (and counting) rules, that may save countless hours of work.

For example, [CombinedAssignRector](https://github.com/rectorphp/rector/blob/master/docs/AllRectorsOverview.md#combinedassignrector), simplifies `$value = $value + 5` into `+$value += 5;`. On Exakat, the rule [Structures/CouldUseShortAssignation]((https://exakat.readthedocs.io/en/latest/Rules.html#could-use-short-assignation) spot those too.

Not all exakat rules are covered by Rector, and vice-versa. [CompactToVariablesRector](https://github.com/rectorphp/rector/blob/master/docs/AllRectorsOverview.md#compacttovariablesrector) aims à skipping usage of compact(), while [Structures/CouldUseCompact](https://exakat.readthedocs.io/en/latest/Rules.html#could-use-compact) suggest the contrary.

Rector and Exakat both use different approaches to code review. It is recommended to review the changes before commiting them.

Check [RectorPHP](https://getrector.org/) website, its [rector github](https://github.com/rectorphp/rector) repository, and [Tomas Votruba](https://twitter.com/VotrubaT) account.


tomas@tomas-ThinkPad-E480: /var...
byzanz-record --d...
tomas@tomas-Th...
tomas@tomas-ThinkPad-E480: /var/www/office-life

```
------------------------------------------------------------------------------------------------
tests/unit/Spaceflow/Service/LocationHttpHeaderGetterTest.php:1
------------------------------------------------------------
Property \Spaceflow\Service\LocationHttpHeaderGetterTest::$stub
(SlevomatCodingStandard\Sniffs\TypeHints\TypeHintDeclarationSni
------------------------------------------------------------

[ERROR] Found 2628 errors that needs to be fixed manually.

→  office-life-backend git:(tv/var-all) damn :/ that will take a
→  office-life-backend git:(tv/var-all)
→  office-life-backend git:(tv/var-all) vendor/bin/rector process src tests
Rector dev-master@b2ed5c7
Config file: /var/www/office-life-backend/rector.yaml

 5075/5313 [▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓░░]  95%
```

```
→ office-life-backend git:(tv/var-all)
grep -roh "@var" src tests | wc -w
1790
→ office-life-backend git:(tv/var-all)
 grep -roh "@var" src tests | wc -w
```

Rector is a Text report format.

Rector depends on the following theme : Rector.

11.4.31 Sarb

The Sarb report is a compatibility report with SARB

SARB is the Static Analysis Results Baseliner. SARB is used to create a baseline of these results. As work on the project progresses SARB can takes the latest static analysis results, removes those issues in the baseline and report the issues raised since the baseline. SARB does this, in conjunction with git, by tracking lines of code between commits. SARB is the brainchild of Dave Liddament.

```
[
    {
        "type": "Classes\/NonPpp",
        "file": "\/home\/exakat\/elation\/code\/include\/base_class.php",
```

(continues on next page)

(continued from previous page)

```
        "line": 37
    },
    {
        "type": "Structures\/NoSubstrOne",
        "file": "\/home\/exakat\/elation\/code\/include\/common_funcs.php",
        "line": 890
    },
    {
        "type": "Structures\/DropElseAfterReturn",
        "file": "\/home\/exakat\/elation\/code\/include\/smarty\/SmartyValidate.class.
↪php",
        "line": 638
    },
    {
        "type": "Variables\/UndefinedVariable",
        "file": "\/home\/exakat\/elation\/code\/components\/ui\/ui.php",
        "line": 174
    },
    {
        "type": "Functions\/TooManyLocalVariables",
        "file": "\/home\/exakat\/elation\/code\/include\/dependencymanager_class.php",
        "line": 43
    }
]
```

Sarb is a Json report format.

Sarb accepts any arbitrary list of results.

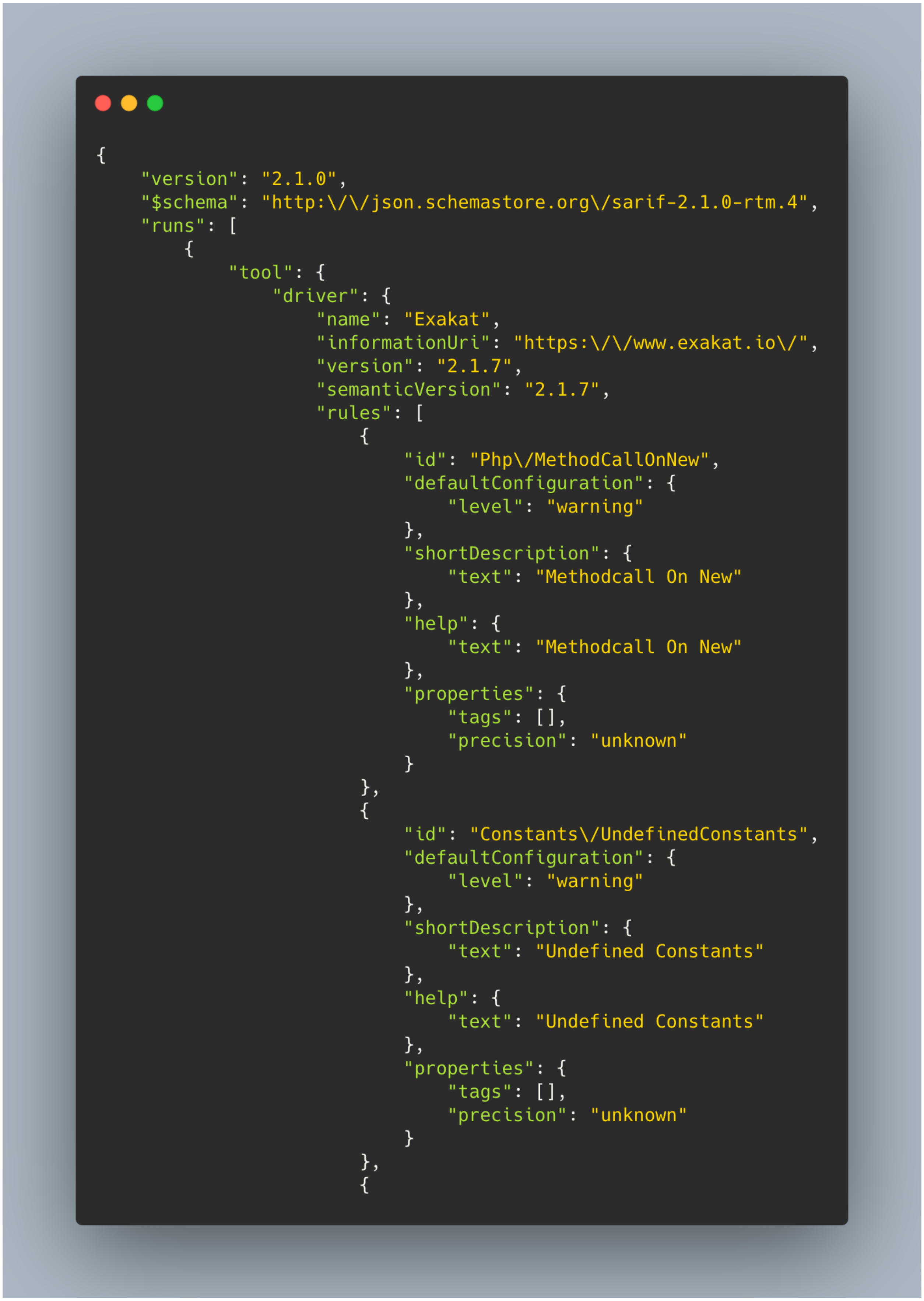
11.4.32 Sarif

The SARIF report publishes the results in SARIF format.

Static Analysis Results Interchange Format (SARIF) a standard format for the output of static analysis tools. The format is referred to as the "Static Analysis Results Interchange Format" and is abbreviated as SARIF.

SARIF is a flexible JSON format, that describes in details the rules, the issues and their context.

More details are available at sarifweb and SARIF support for code scanning at Github.



```
{
    "version": "2.1.0",
    "$schema": "http:\/\/json.schemastore.org\/sarif-2.1.0-rtm.4",
    "runs": [
        {
            "tool": {
                "driver": {
                    "name": "Exakat",
                    "informationUri": "https:\/\/www.exakat.io\/",
                    "version": "2.1.7",
                    "semanticVersion": "2.1.7",
                    "rules": [
                        {
                            "id": "Php\/MethodCallOnNew",
                            "defaultConfiguration": {
                                "level": "warning"
                            },
                            "shortDescription": {
                                "text": "Methodcall On New"
                            },
                            "help": {
                                "text": "Methodcall On New"
                            },
                            "properties": {
                                "tags": [],
                                "precision": "unknown"
                            }
                        },
                        {
                            "id": "Constants\/UndefinedConstants",
                            "defaultConfiguration": {
                                "level": "warning"
                            },
                            "shortDescription": {
                                "text": "Undefined Constants"
                            },
                            "help": {
                                "text": "Undefined Constants"
                            },
                            "properties": {
                                "tags": [],
                                "precision": "unknown"
                            }
                        },
                        {
```

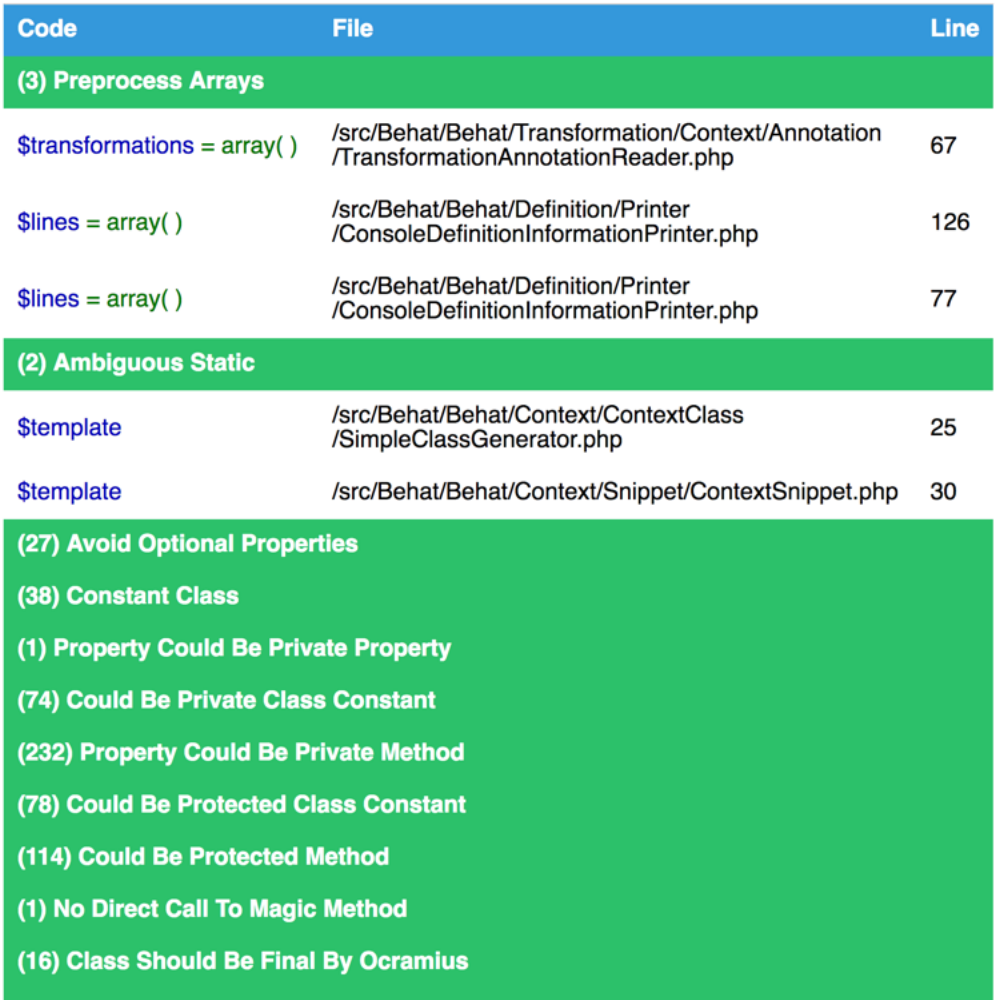
Sarif is a Json report format.

Sarif accepts any arbitrary list of results.

11.4.33 SimpleTable

The Simpletable is a simple table presentation.

Simpletable is suitable for any list of results provided by exakat. It is inspired from the Clang report. The result is a HTML file, with Javascript and CSS.



Code	File	Line
(3) Preprocess Arrays		
$transformations = array( )	/src/Behat/Behat/Transformation/Context/Annotation /TransformationAnnotationReader.php	67
$lines = array( )	/src/Behat/Behat/Definition/Printer /ConsoleDefinitionInformationPrinter.php	126
$lines = array( )	/src/Behat/Behat/Definition/Printer /ConsoleDefinitionInformationPrinter.php	77
(2) Ambiguous Static		
$template	/src/Behat/Behat/Context/ContextClass /SimpleClassGenerator.php	25
$template	/src/Behat/Behat/Context/Snippet/ContextSnippet.php	30
(27) Avoid Optional Properties		
(38) Constant Class		
(1) Property Could Be Private Property		
(74) Could Be Private Class Constant		
(232) Property Could Be Private Method		
(78) Could Be Protected Class Constant		
(114) Could Be Protected Method		
(1) No Direct Call To Magic Method		
(16) Class Should Be Final By Ocramius		

SimpleTable is a HTML report format.

SimpleTable doesn't depend on themes.

11.4.34 Stats

The Stats report collects various stats about the code.

Stats reports PHP structures definition, like class, interfaces, variables, and also features, like operator, control flow instructions, etc.

```
{
    "Summary": {
        "Namespaces": 82,
        "Classes": 59,
        "Interfaces": 29,
        "Trait": 0,
        "Functions": 0,
        "Variables": 4524,
        "Constants": 0
    },
    "Classes": {
        "Classes": 59,
        "Class constants": 10,
        "Properties": 140,
        "Methods": 474
    },
    "Structures": {
        "Ifthen": 568,
        "Else": 76,
        "Switch": 15,
        "Case": 62,
        "Default": 9,
        "Fallthrough": 0,
        "For": 5,
        "Foreach": 102,
        "While": 21,
        "Do..while": 0,
        "New": 106,
        "Clone": 0,
        "Class constant call": 34,
        "Method call": 1071,
        "Static method call": 52,
        "Properties usage": 0,
        "Static property": 65,
        "Throw": 35,
        "Try": 12,
        "Catch": 12,
        "Finally": 0,
        "Yield": 0,
        "Yield From": 0,
        "?  :": 60,
        "?: ": 2,
        "Variables constants": 0,
        "Variables variables": 7,
        "Variables functions": 1,
        "Variables classes": 5
    }
}
```

Stats is a JSON report format.

Stats depends on the following theme : Stats.

11.4.35 Stubs

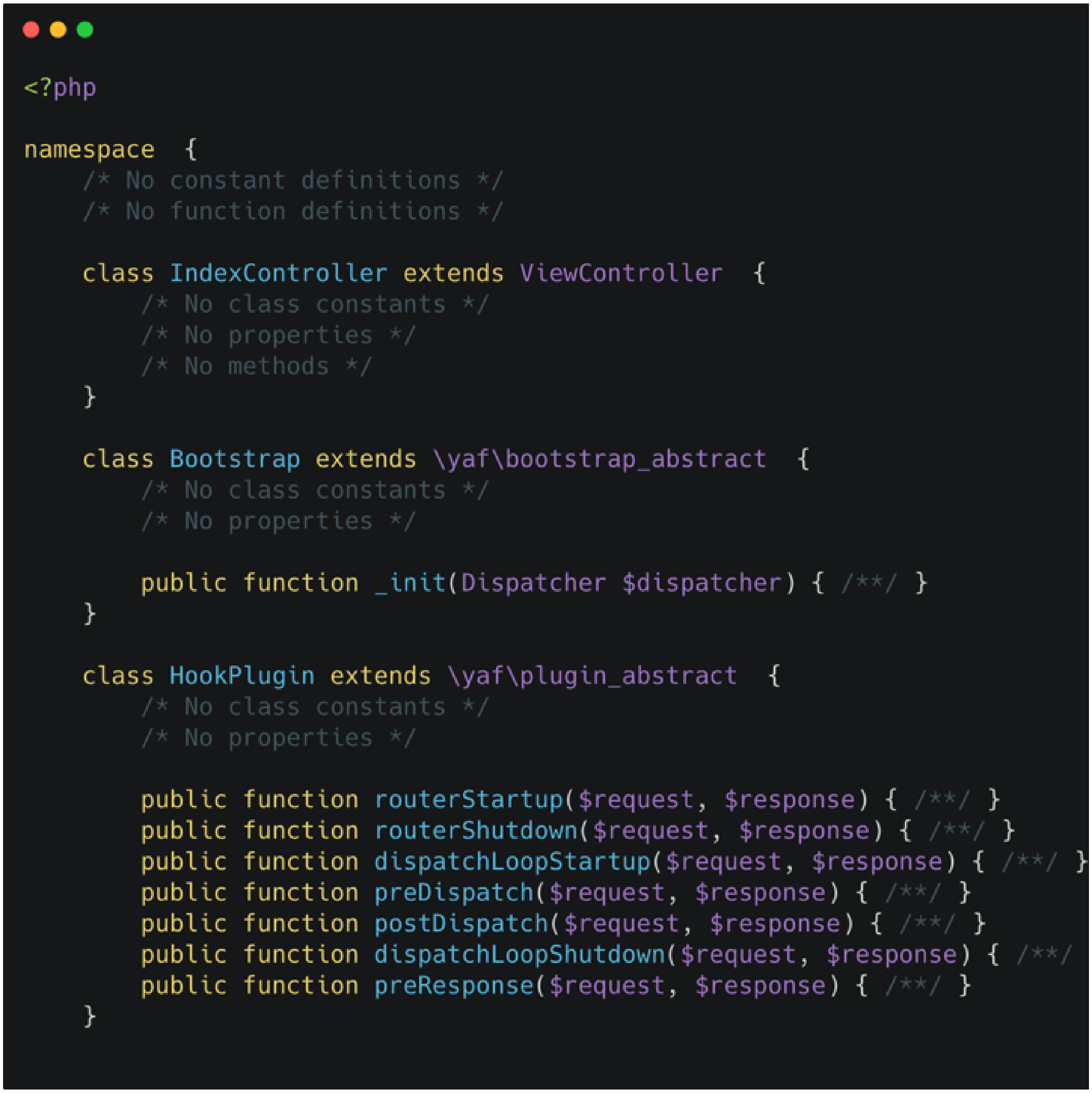
Stubs produces a skeleton from the source code, with all defined structures : constants, functions, classes, interfaces, traits and namespaces.

Stubs takes the original code, and export all defined structures (constants, functions, classes, interfaces, traits and namespaces) in a single and compilable PHP file.

This is convenient for tools that requires documentations for completion, such as IDE.

Constants are exported with their values, properties too. Methods hold their full signature.

The resulting report is in one file, called *stubs.php*.



```
<?php

namespace  {
    /* No constant definitions */
    /* No function definitions */

    class IndexController extends ViewController  {
        /* No class constants */
        /* No properties */
        /* No methods */
    }

    class Bootstrap extends \yaf\bootstrap_abstract  {
        /* No class constants */
        /* No properties */

        public function _init(Dispatcher $dispatcher) { /**/ }
    }

    class HookPlugin extends \yaf\plugin_abstract  {
        /* No class constants */
        /* No properties */

        public function routerStartup($request, $response) { /**/ }
        public function routerShutdown($request, $response) { /**/ }
        public function dispatchLoopStartup($request, $response) { /**/ }
        public function preDispatch($request, $response) { /**/ }
        public function postDispatch($request, $response) { /**/ }
        public function dispatchLoopShutdown($request, $response) { /**/
        public function preResponse($request, $response) { /**/ }
    }
```

Stubs is a PHP report format.

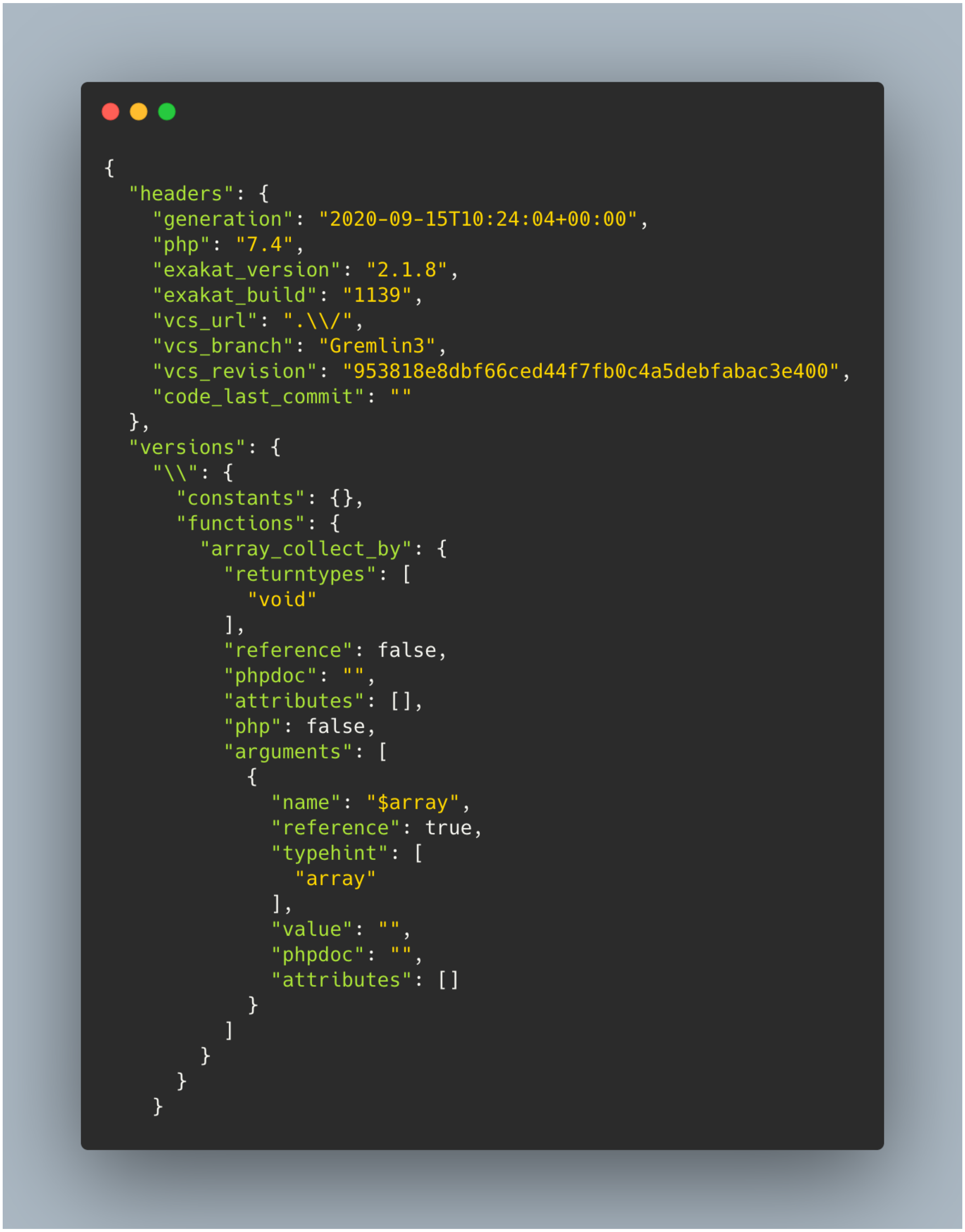
Stubs doesn't depend on themes.

11.4.36 StubsJson

StubsJson produces a complete description of definitions from the code.

The StubsJson report includes :

- Global variables
- Functions
- Constants
- Classes + constants + properties + methods
- Interfaces + constants + methods
- Traits + properties + methods



```
{
  "headers": {
    "generation": "2020-09-15T10:24:04+00:00",
    "php": "7.4",
    "exakat_version": "2.1.8",
    "exakat_build": "1139",
    "vcs_url": ".\\/",
    "vcs_branch": "Gremlin3",
    "vcs_revision": "953818e8dbf66ced44f7fb0c4a5debfabac3e400",
    "code_last_commit": ""
  },
  "versions": {
    "\\": {
      "constants": {},
      "functions": {
        "array_collect_by": {
          "returntypes": [
            "void"
          ],
          "reference": false,
          "phpdoc": "",
          "attributes": [],
          "php": false,
          "arguments": [
            {
              "name": "$array",
              "reference": true,
              "typehint": [
                "array"
              ],
              "value": "",
              "phpdoc": "",
              "attributes": []
            }
          ]
        }
      }
    }
```

StubsJson is a JSON report format.

StubsJson doesn’t depend on themes.

11.4.37 Text

The Text report is a very simple text format.

The Text report displays one result per line, with the following format :

```
/path/from/project/root/to/file:line[space]name of analysis
```

This format is fast, and fitted for machine communications.

```
/classes/test.php:1002      Php/ShouldUseFunction   Should Use Function     array_
↪values(array_unique(array_merge($classTags, $annotations['tags'])))
/classes/test.php:1002      Php/ShouldUseFunction   Should Use Function     array_
↪merge($classTags, $annotations['tags'])
/classes/test.php:1005      Structures/NoArrayUnique        Avoid array_unique()   ␣
↪array_unique(array_merge($classTags, $this->testMethods[$testMethodName]['tags']))
/classes/test.php:1005      Performances/SlowFunctions      Slow Functions  array_
↪unique(array_merge($classTags, $this->testMethods[$testMethodName]['tags']))
```

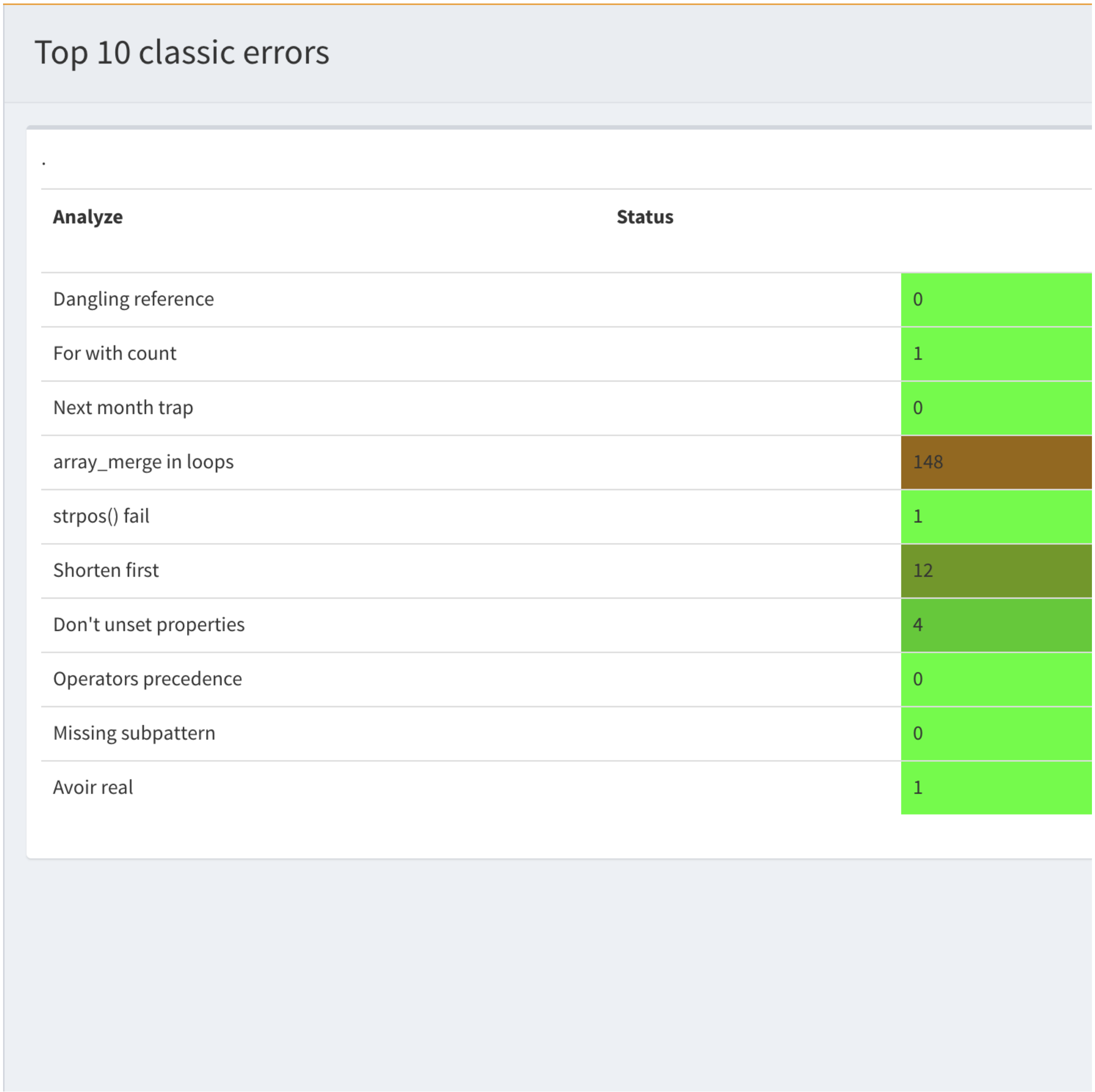
Text is a Text report format.

Text accepts any arbitrary list of results.

11.4.38 Top10

The top 10 is the companion report for the ‘Top 10 classic PHP traps’ presentation.

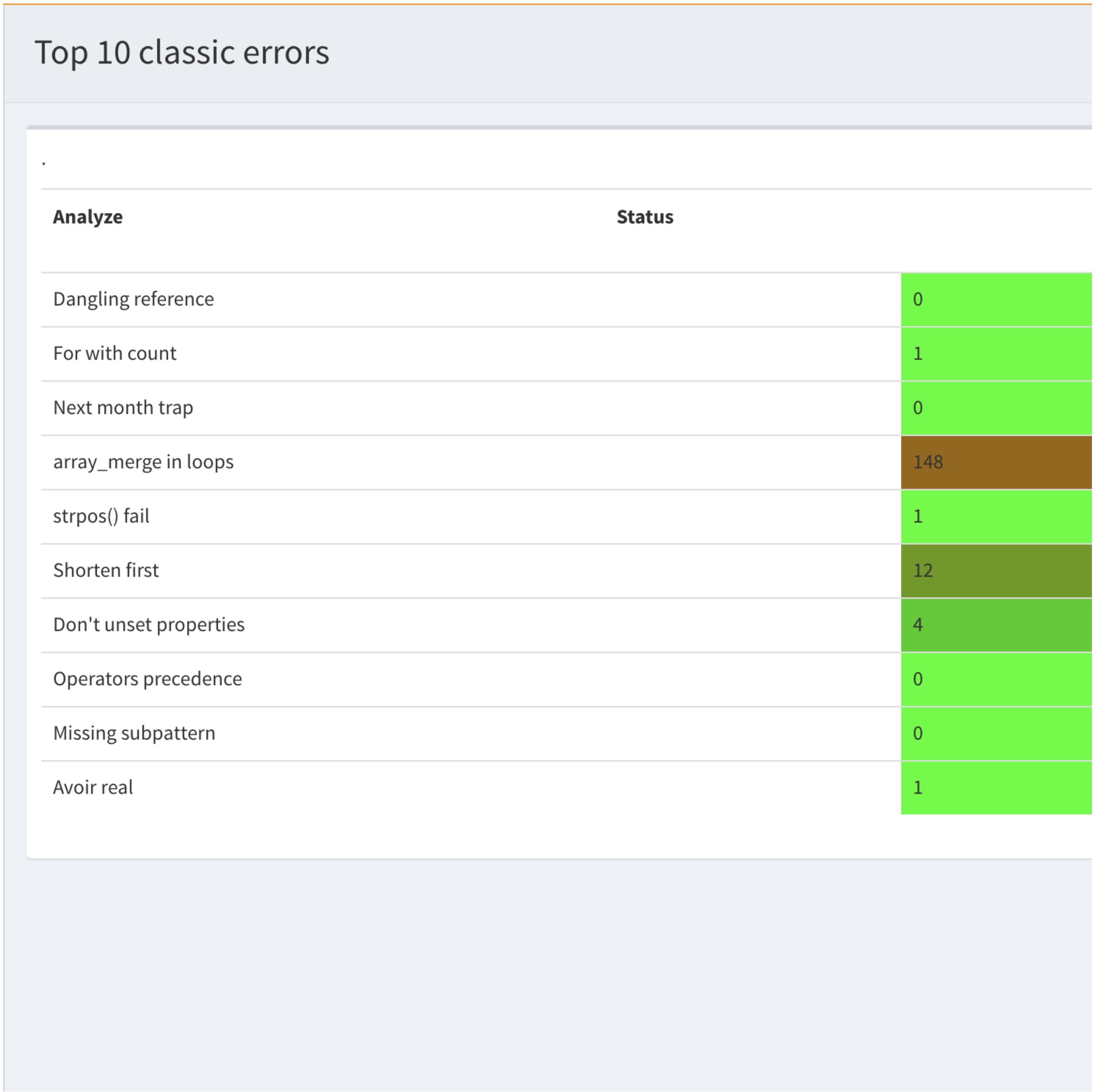
The Top 10 report is based on the ‘Top 10 classic PHP traps’ presentation. You can run it on your code and check immediately where those classic traps are waiting for you. Read the whole presentation online



Top 10 classic errors

.

Analyze	Status
Dangling reference	0
For with count	1
Next month trap	0
array_merge in loops	148
strpos() fail	1
Shorten first	12
Don't unset properties	4
Operators precedence	0
Missing subpattern	0
Avoir real	1



Top 10 classic errors

.

Analyze	Status
Dangling reference	0
For with count	1
Next month trap	0
array_merge in loops	148
strpos() fail	1
Shorten first	12
Don't unset properties	4
Operators precedence	0
Missing subpattern	0
Avoir real	1

Top10 is a HTML report format.

Top10 depends on the following theme : Top10.

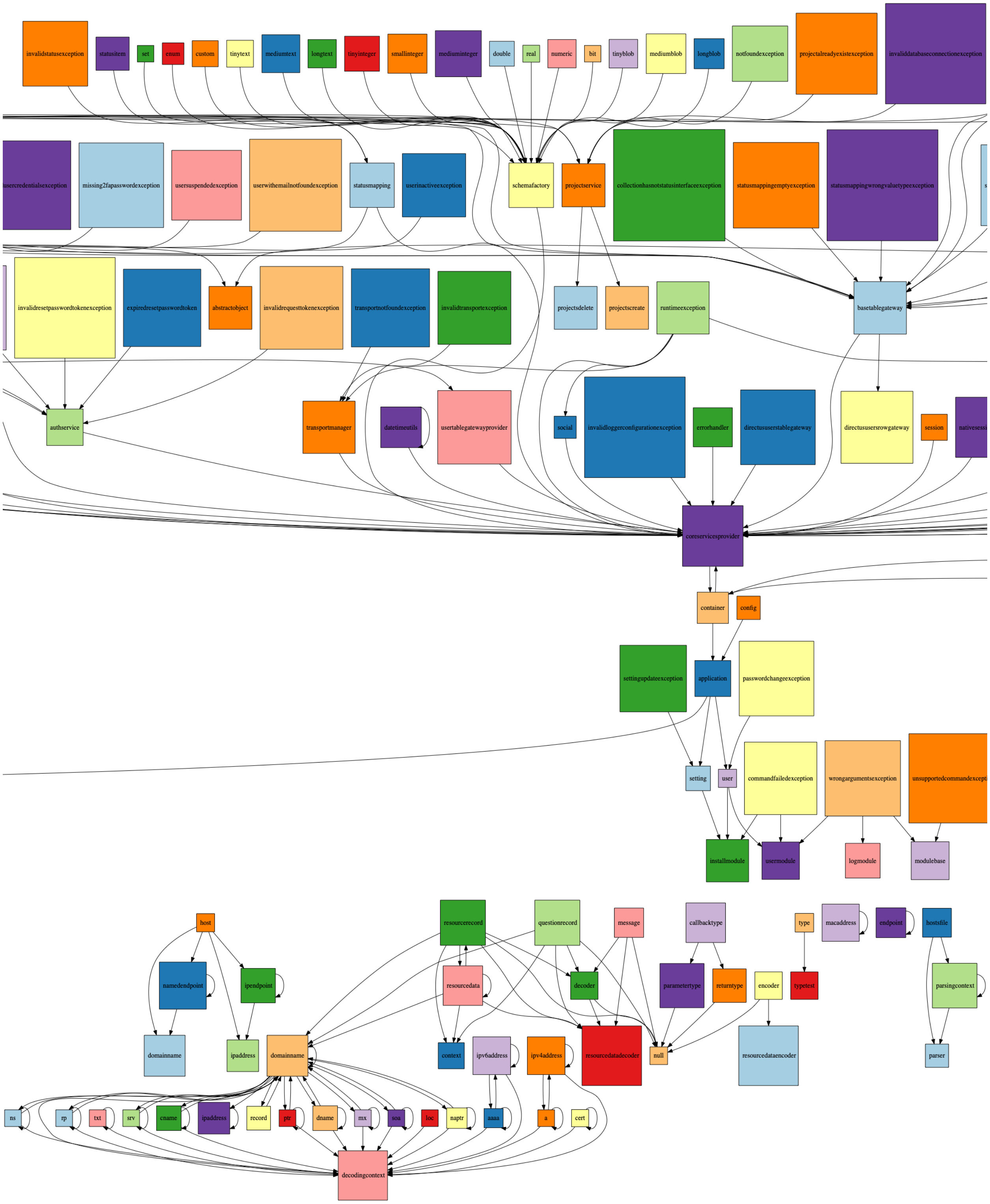
11.4.39 Topology Order

This represents a topological order in the code.

Topology displays all dependencies between code structures. Such dependencies lead to a code hierarchy, which is presented here.

There are currently two topology available:

- Typehint Order : it represents the order in which classes are organized, based on argument and return type.
- New Order : it represents the order in which classes are instantiated, with new.



Topology Order is a DOT report format.

Topology Order doesn't depend on themes.

11.4.40 TypeChecks

The TypeChecks report focuses on reviewing typehint usage.

The TypeChecks report focuses on usage and good usage of typehints.

It checks the presence of typehint, suggests possible type hinting, and check the systemic organisation of the types.

TypeChecks is a HTML report format.

TypeChecks depends on the following theme : TypeChecks.

11.4.41 TypeSuggestion

The TypeSuggestion report provides suggestions to add typehints to methods and properties.

The TypeSuggestion offers suggestions to add typehints to methods and properties.

It provides its suggestion based on the way the code is implemented : by usage or by calling.

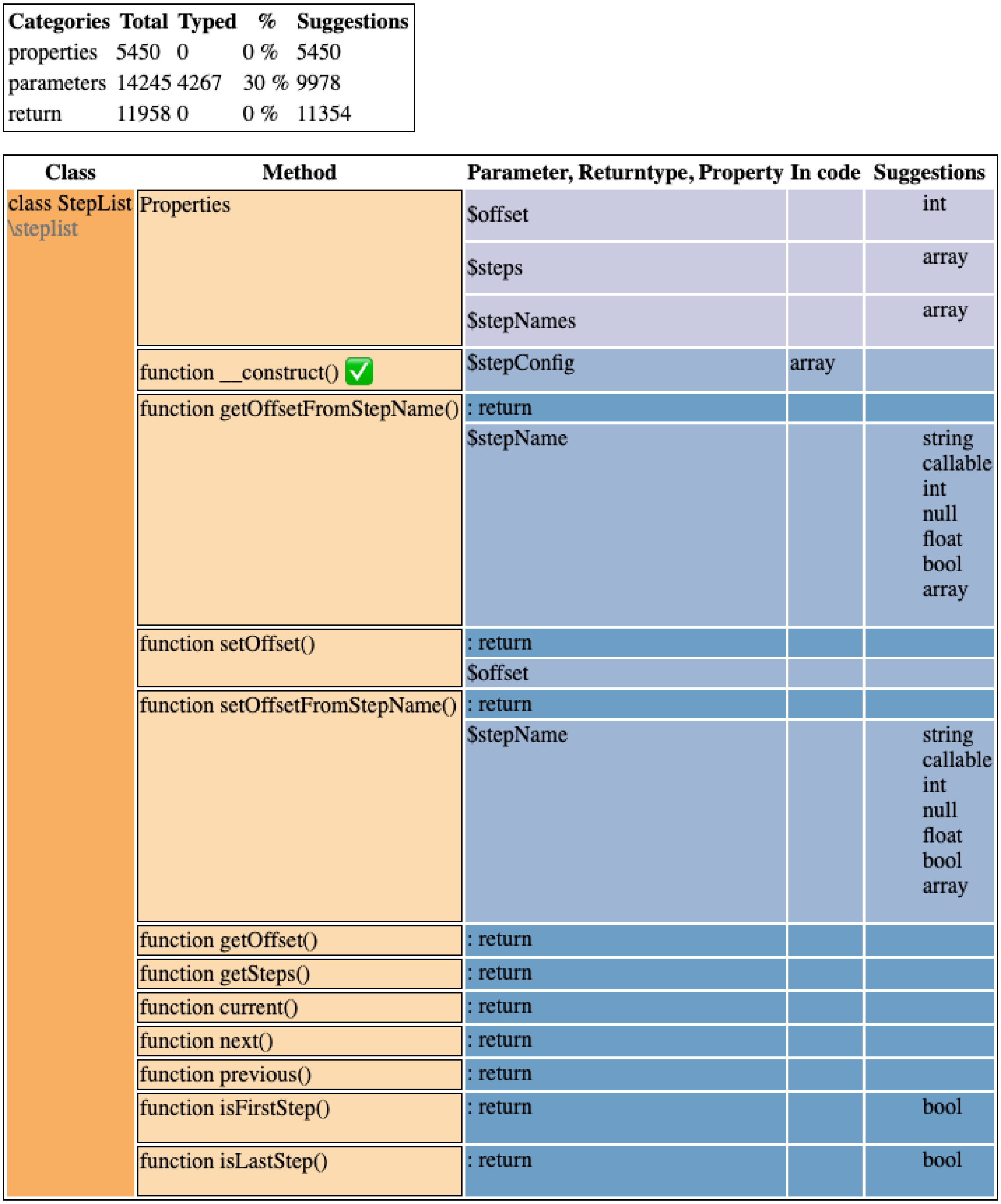
Type usage is the way a typed container is use later. For example, an argument that is used later with the array syntax `$x['a']` or as an object ‘‘$x->b‘‘will receive a suggestion for using array or object.

Type calling is the way the typed container is assigned. For example, a property may receive integer or boolean during assignations : they will receive such suggestions.

Not all types can be guessed : for example, a property may simply hold a value, for later use, such as in a cache system. In such situation, no type is suggested.

`mixed` is not used as suggestion : rather a list of possible types is offered, and it may be upgraded to `mixed`.

This report is ready for PHP 8.0 : the suggestions may be combined together, and multiples suggestions are possible.



Categories	Total	Typed	%	Suggestions
properties	5450	0	0 %	5450
parameters	14245	4267	30 %	9978
return	11958	0	0 %	11354

Class	Method	Parameter, Returntype, Property	In code	Suggestions
class StepList \steplist	Properties	$offset		int
		$steps		array
		$stepNames		array
	function __construct() ✅	$stepConfig	array	
	function getOffsetFromStepName()	: return		
		$stepName		string callable int null float bool array
	function setOffset()	: return		
		$offset		
	function setOffsetFromStepName()	: return		
		$stepName		string callable int null float bool array
	function getOffset()	: return		
	function getSteps()	: return		
	function current()	: return		
	function next()	: return		
	function previous()	: return		
	function isFirstStep()	: return		bool
	function isLastStep()	: return		bool

TypeSuggestion is a HTML report format.

TypeSuggestion depends on the following theme : TypeChecks.

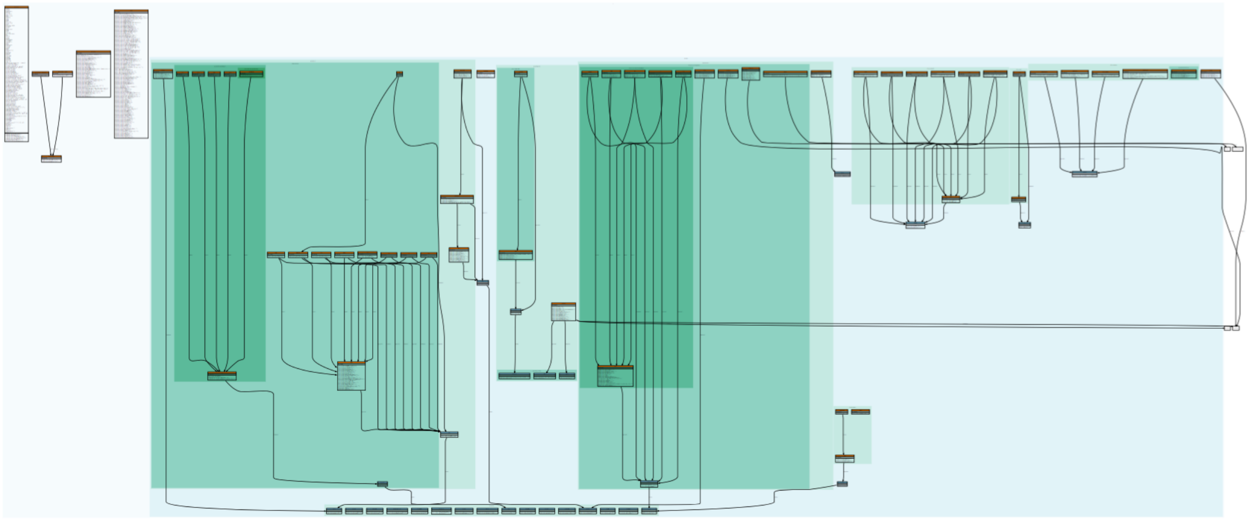
11.4.42 Uml

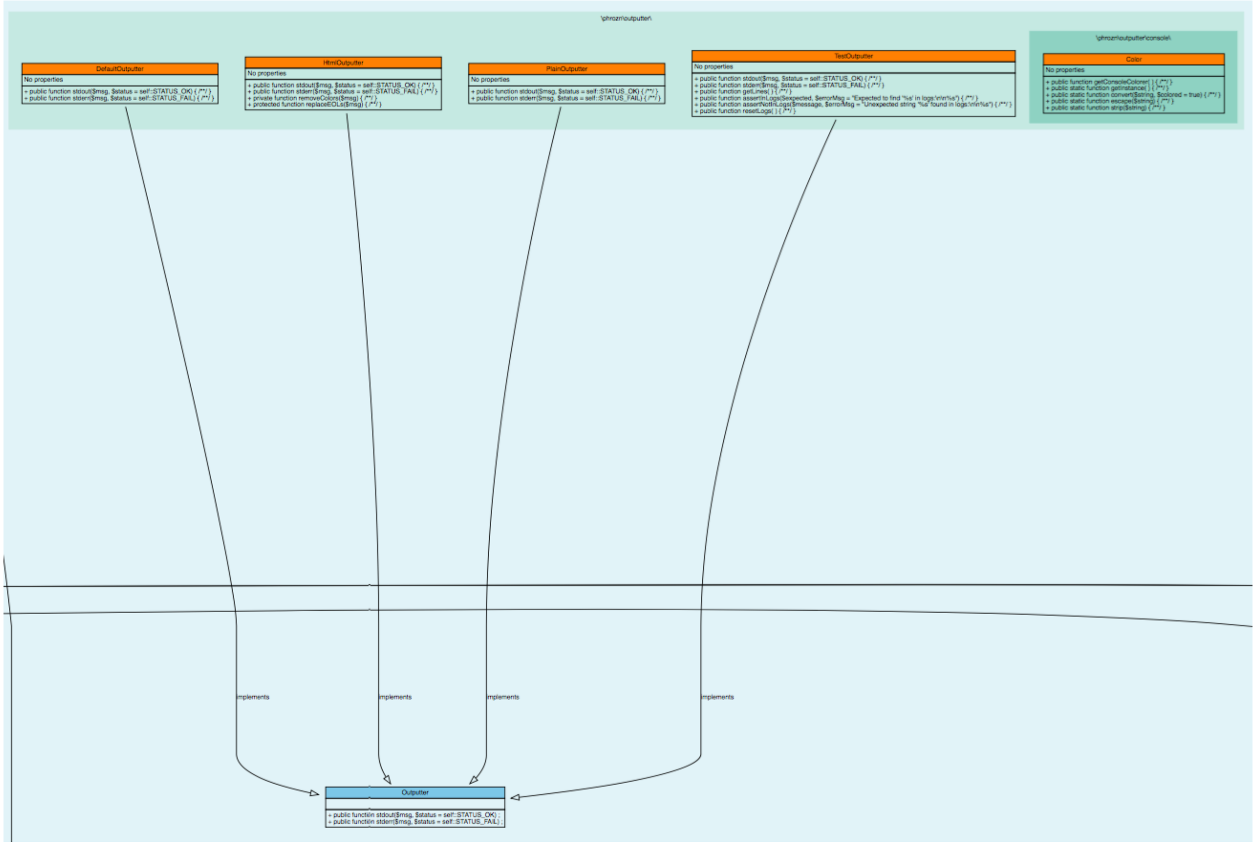
The Uml exports data structure to UML format.

This report produces a dot file with a representation of the classes used in the repository.

Classes, interfaces and traits are represented, along with their constants, methods and properties.

.dot files are best seen with [graphviz](http://www.graphviz.org/) : they are easily convert into PNG or PDF.




\phrozn\outputter\
DefaultOutputter
No properties
+ public function stdout($msg, $status = self::STATUS_OK) { /**/ }
+ public function stderr($msg, $status = self::STATUS_FAIL) { /**/ }
HtmlOutputter
No properties
+ public function stdout($msg, $status = self::STATUS_OK) { /**/ }
+ public function stderr($msg, $status = self::STATUS_FAIL) { /**/ }
+ private function removeColors($msg) { /**/ }
+ protected function replaceEOLs($msg) { /**/ }
PlainOutputter
No properties
+ public function stdout($msg, $status = self::STATUS_OK) { /**/ }
+ public function stderr($msg, $status = self::STATUS_FAIL) { /**/ }
TestOutputter
No properties
+ public function stdout($msg, $status = self::STATUS_OK) { /**/ }
+ public function stderr($msg, $status = self::STATUS_FAIL) { /**/ }
+ public function getLines( ) { /**/ }
+ public function assertInLogs($expected, $errorMsg = "Expected to find '%s' in logs:\n\n%s") { /**/ }
+ public function assertNotInLogs($message, $errorMsg = "Unexpected string '%s' found in logs:\n\n%s") { /**/ }
+ public function resetLogs( ) { /**/ }
\phrozn\outputter\console\
Color
No properties
+ public function getConsoleColorer( ) { /**/ }
+ public static function getInstance( ) { /**/ }
+ public static function convert($string, $colored = true) { /**/ }
+ public static function escape($string) { /**/ }
+ public static function strip($string) { /**/ }
implements
implements
implements
implements
Outputter
+ public function stdout($msg, $status = self::STATUS_OK) ;
+ public function stderr($msg, $status = self::STATUS_FAIL) ;

Uml is a dot report format.

Uml doesn't depend on themes.

11.4.43 Xml

The Xml report exports in XML format.

XML version of the reports. It uses the same format than PHP Code Sniffer to output the results.

```
<?xml version="1.0" encoding="UTF-8"?>
<phpcs version="0.8.6">
<file name="/src/NlpTools/Stemmers/PorterStemmer.php" errors="0" warnings="105"␣
↪fixable="0">
 <warning line="55" column="0" source="Php/EllipsisUsage" severity="Major" fixable="0
↪">... Usage</warning>
```

Xml is a XML report format.

Xml accepts any arbitrary list of results.

11.4.44 Yaml

The Yaml report exports in Yaml format.

Simple Yaml format. It is a structured array with all results, described as object.

```
Filename => [
                errors   => count,
                warning  => count,
                fixable  => count,
                filename => string,
                message  => [
                    line => [
                        type,
                        source,
                        severity,
                        fixable,
                        message
                    ]
                ]
            ]
```

```
/src/Altax/Module/Task/Resource/RuntimeTask.php:
    errors: 0
    warnings: 22
    fixable: 0
    filename: /src/Altax/Module/Task/Resource/RuntimeTask.php
    messages: { 77: [[{ type: warning, source: Structures/Iffectation, severity:␣
↪Minor, fixable: fixable, message: Iffectations, fullcode: '$args = $this->
↪getArguments( )' }]], 67: [[{ type: warning, source: Structures/Iffectation,␣
↪severity: Minor, fixable: fixable, message: Iffectations, fullcode: '$args = $this->
↪input->getArgument(''args'')' }, { type: warning, source: Structures/
↪BuriedAssignation, severity: Minor, fixable: fixable, message: 'Buried Assignation',
↪ fullcode: '$args = $this->input->getArgument(''args'')' }]], 114: [[{ type:␣
↪warning, source: Variables/WrittenOnlyVariable, severity: Minor, fixable: fixable,␣
↪message: 'Written Only Variables', fullcode: $input }, { type: warning, source:␣
↪Variables/VariableUsedOnceByContext, severity: Minor, fixable: fixable, message:
↪'Used Once Variables (In Scope)', fullcode: $input }, { type: warning, source:␣
↪Classes/UndefinedClasses, severity: Major, fixable: fixable, message: 'Undefined␣
↪Classes', fullcode: 'new ArrayInput($arguments)' }]], 13: [[{ type: warning,␣
↪source: Structures/PropertyVariableConfusion, severity: Minor, fixable: fixable,␣
↪message: 'Property Variable Confusion', fullcode: $input }]], 74: [[{ type: warning,
↪ source: Php/ReservedNames, severity: Major, fixable: fixable, message: 'PHP␣
↪Keywords As Names', fullcode: $default }]], 61: [[{ type: warning, source: Php/
↪ReservedNames, severity: Major, fixable: fixable, message: 'PHP Keywords As Names',␣
↪fullcode: $string }]], 59: [[{ type: warning, source: Php/ReservedNames, severity:␣
↪Major, fixable: fixable, message: 'PHP Keywords As Names', fullcode: $string }, {␣
↪type: warning, source: Functions/RelayFunction, severity: Major, fixable: fixable,␣
```

(continues on next page)

(continued from previous page)

Yaml is a Yaml report format.

Yaml accepts any arbitrary list of results.

CHAPTER 12

Configuration

12.1 Summary

12.2 Common Behavior

12.2.1 General Philosophy

Exakat tries to avoid configuration as much as possible, so as to focus on working out of the box, rather than spend time on pre-requisite.

As such, it probably does more work, but that may be dismissed later, at reading time.

More configuration options appear with the evolution of the engine.

12.2.2 Precedence

The exakat engine read directives from three places :

1. The command line options

2. The .exakat.ini file at the root of the code
3. The config.ini file in the project directory
4. The exakat.ini file in the config directory
5. The default values in the code

The precedence of the directives is the same as the list above : command line options always have highest priority, config.ini files are in second, when command line are not available, and finally, the default values are read in the code.

Some of the directives are only available in the config.ini files.

12.2.3 Common Options

All options are the same, whatever the command provided to exakat. -f always means files, and -q always means quick.

Any option that a command doesn't understand is ignored.

Any option that is not recognized is ignored and reported (with visibility).

12.3 Engine configuration

Engine configuration is were the exakat engine general configuration are stored. For example, the php binaries or the neo4j folder are there. Engine configurations affect all projects.

12.3.1 Configuration File

The Exakat engine is configured in the 'config/exakat.ini' file.

This file is created with the 'doctor' command, or simply by copying another such file from another installation.

```
php exakat.phar doctor
```

When the doctor can't find the 'config/config.ini' file, it attempts to create one, with reasonable values. It is recommended to use this to create the exakat.ini skeleton, and later, modify it.

12.3.2 Available Options

Here are the currently available options in Exakat's configuration file : config/config.ini

Option	Description
graphdb	The graph database to use. Currently, it may be gsneo4j, or tinkergraph.
gsneo4j_host	The host to connect to reach the graph database, when using gsneo4j driver. The default value is 'localhost'
gsneo4j_host	The port to use on the host to reach the graph database, when using gsneo4j driver.. The default value is '8182
gsneo4j_folder	The folder where the code for the graph database resides, when using gsneo4j driver. The default value is 'tinl
tinkergraph_host	The host to connect to reach the graph database, when using tinkergraph driver. The default value is 'localhos
tinkergraph_port	The port to use on the host to reach the graph database, when using tinkergraph driver. The default value is '8
tinkergraph_folder	The folder where the code for the graph database resides, when using tinkergraph driver. The default value is
gsneo4jv3_host	The host to connect to reach the graph database, when using gsneo4j driver. The default value is 'localhost'
gsneo4jve_host	The port to use on the host to reach the graph database, when using gsneo4j driver.. The default value is '8182

Option	Description
gsneo4jv3_folder	The folder where the code for the graph database resides, when using gsneo4j driver. The default value is 'tinl
tinkergraphv3_host	The host to connect to reach the graph database, when using tinkergraph driver. The default value is 'localhos
tinkergraphv3_port	The port to use on the host to reach the graph database, when using tinkergraph driver. The default value is '8
tinkergraphv3_folder	The folder where the code for the graph database resides, when using tinkergraph driver. The default value is
project_rulesets	List of analysis rulesets to be run. The list may include extra rulesets that are not used by the default reports :
project_themes	Obsolete. Use the one above : project_rulesets
project_reports	The list of reports that can be produced when running 'project' command. This list may automatically add ext
token_limit	Maximum size of the analyzed project, in number of PHP tokens (excluding whitespace). Use this to avoid ru
php	Link to the PHP binary. This binary is the one that runs Exakat. It is recommended to use PHP 7.3, or 7.4. Th
php80	Path to the PHP 8.0.x binary. This binary is needed to test the compilation with the 8.0 series or if the analyze
php74	Path to the PHP 7.4.x binary. This binary is needed to test the compilation with the 7.4 series or if the analyze
php73	Path to the PHP 7.3.x binary. This binary is needed to test the compilation with the 7.3 series or if the analyze
php72	Path to the PHP 7.2.x binary. This binary is needed to test the compilation with the 7.2 series or if the analyze
php71	Path to the PHP 7.1.x binary. This binary is needed to test the compilation with the 7.1 series or if the analyze
php70	Path to the PHP 7.0.x binary. This binary is needed to test the compilation with the 7.0 series or if the analyze
php56	Path to the PHP 5.6.x binary. This binary is needed to test the compilation with the 5.6 series or if the analyze
php55	Path to the PHP 5.5.x binary. This binary is needed to test the compilation with the 5.5 series or if the analyze
php54	Path to the PHP 5.4.x binary. This binary is needed to test the compilation with the 5.4 series or if the analyze
php53	Path to the PHP 5.3.x binary. This binary is needed to test the compilation with the 5.3 series or if the analyze
php52	Path to the PHP 5.2.x binary. This binary is needed to test the compilation with the 5.2 series or if the analyze
php_extensions	List of PHP extensions to use when spotting functions, methods, constants, classes, etc. Default to 'all', which

Note : php** configuration may be either a valid PHP binary path, or a valid Docker image. The path on the system may be */usr/bin/php*, */usr/sbin/php80*, or */usr/local/Cellar/php71/7.1.30/bin/php*. The Docker configuration must have the form *abc/def:tag*. The image's name may be any value, as long as Exakat manage to run it, and get the valid PHP signature, with *php -v*. When using Docker, the docker server must be running.

12.3.3 Custom rulesets

Create custom rulesets by creating a 'config/themes.ini' directive files.

This file is a .INI file, build with several sections. Each section is the name of a ruleset : for example, 'mine' is the name for the ruleset below.

There may be several sections, as long as the names are distinct.

It is recommended to use all low-case names for custom rulesets. Exakat uses names with a first capital letter, which prevents conflicts. Behavior is undefined if a custom ruleset has the same name as a default ruleset.

```
['mine']
analyzer[] = 'Structures/AddZero';
analyzer[] = 'Performances/ArrayMergeInLoops';
```

The list of analyzer in the ruleset is based on the 'analyzer' array. The analyzer is identified by its 'shortname'. Analyzer shortname may be found in the documentation (*Rules list* or within the Ambassador report). Analyzers names have a 'A/B' structure.

The list of available rulesets, including the custom ones, is listed with the *doctor* command.

12.4 Project Configuration

Project configuration are were the project specific configuration are stored. For example, the project name, the ignored directories or its external libraries are kept. Configurations only affect one project and not the others.

Project configuration file are called 'config.ini'. They are located, one per project, in the 'projects/&lt;project name&gt;/config.ini' file.

12.4.1 Available Options

Here are the currently available options in Exakat's project configuration file : projects/&lt;project name&gt;/config.ini

Option	Description
phpversion	Version with which to run the analyze. It may be one of : 7.3, 7.2, 7.1, 7.0, 5.6, 5.5, 5.4, 5.3, 5.2. Default is 7.2 or the CLI version used to init the project. 5.* versions are available, but are less tested. 7.3 is actually the current dev version.
include_dirs[]	This is the list of files and dir to include in the project's directory. It is chrooted in the project's folder. Values provided with a starting / are used as a path prefix. Values without / are used as a substring, anywhere in the path. include_dirs are added AFTER ignore_dirs, so as to partially ignore a folder, such as the vendor folder from composer.
ignore_dirs[]	This is the list of files and dir to ignore in the project's directory. It is chrooted in the project's folder. Values provided with a starting / are used as a path prefix. Values without / are used as a substring, anywhere in the path.
ignore_dirs[]	This is the list of files and dir to ignore in the project's directory. It is chrooted in the project's folder. Values provided with a starting / are used as a path prefix. Values without / are used as a substring, anywhere in the path.
file_extensions	This is the list of file extensions that is considered as PHP scripts. All others are ignored. All files bearing those extensions are subject to check, though they are scanned first for PHP tags before being analyzed. The extensions are comma separated, without dot. The default are : php, php3, inc, tpl, phtml, tmpl, phps, ctp
project_name	This is the project name, as it appears at the top left in the Ambassador report.
project_url	This is the repository URL for the project. It is used to get the source for the project.
project_vcs	This is the VCS used to fetch the project source.
project_description	This is the description of the project.
project_packagist	This is the packagist name for the code, when the code is fetched with composer.

12.5 Adding/Excluding files

ignore_dirs and include_dirs are the option used to select files within a folder. Here are some tips to choose

- From the full list of files, ignore_dirs[] is applied, then include_dirs is applied. The remaining list is processed.
- ignore one file : *ignore_dirs[] = "/path/to/file.php"*
- ignore one dir : *ignore_dirs[] = "/path/to/dir/"*
- ignore siblings but include one dir : *ignore_dirs[] = "/path/to/parent/"; include_dirs[] = "/path/to/parent/dir/"*
- ignore every name containing 'test' : *ignore_dirs[] = "test";*
- only include one dir (and exclude the rest): *include_dirs[] = "/path/to/dir/";*
- omitting include_dirs defaults to *"include_dirs[] = ""*

- omitting ignore_dirs defaults to *"ignore_dirs[] = ""*
- including or ignoring files multiple times only has effect once

include_dirs has priority over the *config.cache* configuration file. If a folder has been marked for exclusion in the *config.cache* file, it may be forced to be included by configuring its value with include_dirs in the *config.ini* file.

12.6 In-code Configuration

In-code configuration is a configuration file that sits at the root of the code. When exakat finds it, it uses it for in-code auditing.

The file is *.exakat.yaml*, and is a valid YAML file. *.exakat.yml* is also valid, but not recommended.

In case the file is found but not valid, Exakat reverts to default values.

Unrecognized values are ignored.

12.6.1 Exakat in-code example

```
project: exakat
project_name: exakat
project_rulesets:
- my_ruleset
- Security
project_report:
- Ambassador
file_extensions: php,php3,phtml
include_dirs:
  - /
ignore_dirs:
  - /tests
  - /vendor
  - /docs
  - /media
ignore_rules:
  - Structures/AddZero
rulesets:
  my_ruleset:
      - Structures/AddZero
      - Structures/MultiplyByOne
```

12.6.2 Exakat in-code skeleton

Copy-paste this YAML code into a file called *.exakat.yaml*, located at the root of your repository.

```
file_extensions: php,php3,phtml
project: <project short name>
project_name: <project name, as displayed in reports>
project_rulesets:
- <list of rulesets to apply>
- Analysis
file_extensions: php,php3,phtml
project_report:
```

(continues on next page)

(continued from previous page)

```
- <list of reports to build>
- Ambassador
include_dirs:
  - /
ignore_rules:
  -
ignore_dirs:
  - /tests
  - /vendor
  - /docs
  - /media
```

12.6.3 Available Options

Here are the currently available options in Exakat's project configuration file : projects/&lt;project name&gt;/config.ini

Option	Description
include_dirs[]	This is the list of files and dir to include in the project's directory. It is chrooted in the project's folder. Values provided with a starting / are used as a path prefix. Values without / are used as a substring, anywhere in the path. include_dirs are added AFTER ignore_dirs, so as to partially ignore a folder, such as the vendor folder from composer.
ignore_dirs[]	This is the list of files and dir to ignore in the project's directory. It is chrooted in the project's folder. Values provided with a starting / are used as a path prefix. Values without / are used as a substring, anywhere in the path.
ignore_rules[]	The rules mentioned in this list are ignored when running the audit. Rules are ignored after loading the rulesets configuration : as such, it is possible to ignore rules inside a ruleset, without ignoring the whole ruleset. The rules in this list are Exakat's short name : ignore_rules[] = "Structures/AddZero"
file_extensions	This is the list of file extensions that is considered as PHP scripts. All others are ignored. All files bearing those extensions are subject to check, though they are scanned first for PHP tags before being analyzed. The extensions are comma separated, without dot. The default are : php, php3, inc, tpl, phtml, tmpl, phps, ctp
project_name	This is the project name, as it appears at the top left in the Ambassador report.
project_url	This is the repository URL for the project. It is used to get the source for the project.
project_vcs	This is the VCS used to fetch the project source.
project_description	This is the description of the project.
project_packagist	This is the packagist name for the code, when the code is fetched with composer.

12.7 Commandline Configuration

Commandline configurations are detailled with each command, in the _Commands section.

12.8 Specific analysis configurations

Some analyzer may be configured individually. Those parameters are then specific to one analyzer, and it only affects their behavior.

Analyzers may be configured in the *project/*/config.ini*; they may also be configured globally in the *config/exakat.ini* file.

Array() / [ ] Consistence

- array_ratio : 10
  - Percentage of arrays in one of the syntaxes, to trigger the other syntax as a violation.

Too Many Array Dimensions

- maxDimensions : 3
  - Number of valid dimensions in an array.

Custom Class Usage

- forbiddenClasses :
  - List of classes to be avoided

Cancel Common Method

- cancelThreshold : 75
  - Minimal number of cancelled methods to suggest the cancellation of the parent.

Could Be Parent Method

- minChildren : 4
  - Minimal number of children using this method.

Fossilized Method

- fossilizationThreshold : 6
  - Minimal number of overwriting methods to consider a method difficult to update.

Make Magic Concrete

- magicMemberUsage : 1
  - Minimal number of magic member usage across the code, to trigger a concrete property.

Too Many Children

- childrenClassCount : 15
  - Threshold for too many children classes for one class.

Too Many Dereferencing

- tooManyDereferencing : 7
  - Maximum number of dereferencing.

Too Many Finds

- minimumFinds : 5
  - Minimal number of prefixed methods to report.

Too Many Finds

- findPrefix : find
  - list of prefix to use when detecting the 'find'. Comma-separated list, case insensitive.

Too Many Injections

- injectionsCount : 5
  - Threshold for too many injected parameters for one class.

Large Try Block

- tryBlockMaxSize : 5
  - Maximal number of expressions in the try block.

Missing Include

- constant_or_variable_name : 100
  - Literal value to be used when including files. For example, by configuring 'Files_MissingInclude["HOME_DIR"] = "/tmp/myDir/";', then 'include HOME_DIR . "my_class.php"; will be actually be used as '/tmp/myDir/my_class.php'. Constants must be configured with their correct case. Variable must be configured with their initial '$'. Configure any number of variable and constant names.

Could Make A Function

- centralizeThreshold : 8
  - Minimal number of calls of the function with one common argument.

Hardcoded Passwords

- passwordsKeys : password_keys.json
  - List of array index and property names that shall be checked for potential secret key storages.

Prefix And Suffixes With Typehint

- prefixedType : prefixedType['is'] = 'bool';

prefixedType['has'] = 'bool'; prefixedType['set'] = 'void'; prefixedType['list'] = 'array';

- List of prefixes and their expected returntype

Prefix And Suffixes With Typehint

- suffixedType : prefixedType['list'] = 'bool';

prefixedType['int'] = 'int'; prefixedType['string'] = 'string'; prefixedType['name'] = 'string'; prefixedType['description'] = 'string'; prefixedType['id'] = 'int'; prefixedType['uuid'] = 'Uuid';

- List of suffixes and their expected returntype

Too Many Local Variables

- tooManyLocalVariableThreshold : 15
  - Minimal number of variables in one function or method to report.

Too Many Parameters

- parametersCount : 8
  - Minimal number of parameters to report.

Too Much Indented

- indentationAverage : 1
  - Minimal average of indentation in a method to report. Default is 1.0, which means that the method is on average at one level of indentation or more.

Too Much Indented

- minimumSize : 3
  - Minimal number of expressions in a method to apply this analysis.

Abstract Away

- abstractableCalls :
    - Functions that shouldn't be called directly, unless in a method.

Abstract Away

- abstractableClasses :
    - Classes that shouldn't be instantiated directly, unless in a method.

Memoize MagicCall

- minMagicCallsToGet : 2
    - Minimal number of calls of a magic property to make it worth locally caching.

PHP Keywords As Names

- reservedNames :
    - Other reserved names : all in a string, comma separated.

PHP Keywords As Names

- allowedNames :
    - PHP reserved names that can be used in the code. All in a string, comma separated.

Too Many Native Calls

- nativeCallCounts : 3
    - Number of native calls found inside another call.

Keep Files Access Restricted

- filePrivileges : 0777
    - List of forbidden file modes (comma separated).

Should Use Prepared Statement

- queryMethod : query_methods.json
    - Methods that call a query.

Long Arguments

- codeTooLong : 100
    - Minimum size of a functioncall or a methodcall to be considered too long.

Too Long A Block

- longBlock : 200
    - Size of a block for it to be too long. A block is commanded by a for, foreach, while, do…while, if/then else structure.

Max Level Of Nesting

- maxLevel : 4
    - Maximum level of nesting for control flow structures in one scope.

Nested Ifthen

- nestedIfthen : 3

  - Maximal number of acceptable nesting of if-then structures

@ Operator

- authorizedFunctions : noscream_functions.json
  - Functions that are authorized to sports a @.

Duplicate Literal

- minDuplicate : 15
  - Minimal number of duplication before the literal is reported.

12.9 Configuring analysis to be run

Exakat builds a list of analysis to run, based on two directives : *project_reports* and *projects_themes*. Both are list of rulesets. Unknown rulesets are omitted.

project_reports makes sure you can extract those reports, while *projects_themes* allow you to build reports a la carte later, and avoid running the whole audit again.

12.9.1 Required rulesets

First, analysis are very numerous, and it is very tedious to sort them by hand. Exakat only handles 'themes' which are groups of analysis. There are several list of rulesets available by default, and it is possible to customize those lists.

When using the *projects_themes* directive, you can configure which rulesets must be processed by exakat, each time a 'project' command is run. Those rulesets are always run.

12.9.2 Report-needed rulesets

Reports are build based on results found during the auditing phase. Some reports, like 'Ambassador' or 'Drillinstructor' needs the results of specific rulesets. Others, like 'Text' or 'Json' build reports at the last moment.

As such, exakat uses the project_reports directive to collect the list of necessary rulesets, and add them to the *projects_themes* results.

12.9.3 Late reports

It is possible de extract a report, even if the configuration has not been explicitly set for it.

For example, it is possible to build the Owasp report after telling exakat to build a 'Ambassador' report, as Ambassador includes all the analysis needed for Owasp. On the other hand, the contrary is not true : one can't get the Ambassador report after running exakat for the Owasp report, as Owasp only covers the security rulesets, and Ambassador requires other rulesets.

12.9.4 Recommendations

- The 'Ambassador' report has all the classic rulesets, it's the most comprehensive choice.
- To collect everything possible, use the ruleset 'All'. It's also the longest-running ruleset of all.
- To get one report, simply configure project_report with that report.

- You may configure several rulesets, like ‘Security’, ‘Suggestions’, ‘CompatibilityPHP73’, and later extract independant results with the ‘Text’ or ‘Json’ format.
- If you just want one compulsory report and two optional reports (total of three), simply configure all of them with project_report. It’s better to produce extra reports, than run again a whole audit to collect missing informations.
- It is possible to configure customized rulesets, and use them in project_rulesets
- Excluding one analyzer is not supported. Use custom rulesets to build a new one instead.

12.9.5 Example

```
project_reports[] = 'Drillinstructor';
project_reports[] = 'Owasp';

project_themes[] = 'Security';
project_themes[] = 'Suggestions';
```

With that configuration, the Drillinstructor and the Owasp report are created automatically when running ‘project’. Use the following command to get the specific rulesets ;

```
php exakat.phar report -p <project> -format Text -T Security -v
```

12.10 Check Install

Once the prerequisite are installed, it is advised to run to check if all is found :

php exakat.phar doctor

After this run, you may edit ‘config/config.ini’ to change some of the default values. Most of the time, the default values will be OK for a quick start.

CHAPTER 13

Custom analysis

13.1 Summary:

13.2 How Exakat runs an analysis

An analysis is the smallest unit of work on the Exakat engine.

An analysis is constituted of several elements :

- A name, including a prefix called ‘folder’.
- A class, that extends *ExakatAnalyzerAnalyzer*.
- A documentation

- Unit tests

The exakat command *analyze* runs an analysis, either alone, or as a member of one category. An analysis may be part of multiple categories. Categories gathers several analysis together, to be used by a report.

```
# run one analysis alone
php exakat analyze -p test -P Structures/AddZero

# run an analysis as apart of of a category : Structures/AddZero belongs to Analyze
php exakat analyze -p test -T Analyze
```

An analysis can only be run after loading the code in the central database, with the 'load' command. It is then 'dump'-ed before being reported as an audit.

The prefix is used for internal identification and storage. It is unique.

13.3 Quick startup

To create a new analysis, you must work with the Exakat source code. Start by cloning the repository :

`git clone exakat https://github.com/exakat/exakat.git`

Then move to the cloned directory. Here, call the following script to create an analysis :

`php scripts/createAnalyzer Custom/FirstTest`

This script creates the following files :

- library/Exakat/Analyze/Custom/FirstTest.php
- human/en/Custom/FirstTest.ini
- tests/analyzer/Test/Custom/FirstTest.php
- tests/analyzer/source/Custom/FirstTest.01.php
- tests/analyzer/exp/Custom/FirstTest.01.php
- It also updates a file called *data/analyzers.sqlite*

Open *library/Exakat/Analyze/Custom/FirstTest.php* in your favorite IDE. The code looks like the following :

```
<?php
/*
 * Copyright 2012-2018 Damien Seguy - Exakat Ltd <contact(at)exakat.io>
 * This file is part of Exakat.
 *
 * Exakat is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Affero General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * Exakat is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU Affero General Public License for more details.
 *
 * You should have received a copy of the GNU Affero General Public License
 * along with Exakat.  If not, see <http://www.gnu.org/licenses/>.
 *
```

(continues on next page)

(continued from previous page)

```
 * The latest code can be found at <http://exakat.io/>.
 *
*/

namespace Exakat\Analyzer\Custom;

use Exakat\Analyzer\Analyzer;

class FirstTest extends Analyzer {
    /* Remove this if useless
    public function dependsOn() {
        return array('MethodDefinition');
    }
    */

    public function analyze() {
        $this->atomIs('Identifier')
             ->back('first');
        $this->printQuery();
        $this->prepareQuery();
    }
}

?>
```

The main part of the analysis is the *analyze* method, so we'll focus on it. It has a very simple example code, and runs a debugging tool. Let start by removing the line *$this->printQuery();*.

For this example, we're going to look for useless additions : something like *$a + 0*. For that, we need to detect additions, check that the operator is + and find if one of the operand is 0. For that, we're going to replace the first call to *$this* by the following code :

```
$this->atomIs('Addition')
     ->codeIs('+')
     ->outIs('RIGHT')
     ->atomIs('Integer')
     ->codeIs('0');
$this->prepareQuery();
```

The query may be read as the following : find all atoms that are 'Addition', check if the code is '+', then go to the 'RIGHT' expression, check if the atom is an 'Integer', then check if the value is '0'. If all of those steps are valid, the resulting element is now a result for our analysis.

The next call to 'prepareQuery' means that this query is complete.

At that point, our analysis is build with one query. It is executed by the exakat engine.

Now, we need to start testing our analysis and check if all works as expected. The simplest is to rely on the unit tests to validate the analysis.

Open the file *tests/analyzer/source/Custom/MyFirst.01.php*. Inside, you'll find an empty PHP script.

```
<?php

?>
```

Let's complete this script with code that we intend to analyse. For that, we simply PHP code that hold the pattern we are looking for. For example :

```
<?php

$a + 0;
0 + $a;

1 + $a;
$a + 1;
?>
```

As you can see, we added patterns of code that we would like to find, like *$a + 0* and *0 + $a*. We also added patterns of code we don’t want to find, like *$a + 1* and *1 + $a*. It is important to tell the tests what we expect, and what we want to avoid.

Save that file, and open the next one : *tests/analyzer/exp/Custom/MyFirst.01.php*.

```
<?php

$expected     = array('',
                      '',
                     );

$expected_not = array('',
                      '',
                     );

?>
```

This file holds the expected values and the values we want to avoid. The expected values are *$a + 0* and *0 + $a*, so added them in the *$expected* array. The unwanted values are *$a + 1* and *1 + $a*, so added them in the *$expected_not* array.

```
<?php

$expected     = array('$a + 0',
                      '0 + $a',
                     );

$expected_not = array('$a + 1',
                      '1 + $a',
                     );

?>
```

Save the file too. We are now ready to run this test with PHPunit. Check that PHPunit is installed, then run the test.

```
cd tests/analyzer/
phpunit Test/Custom/MyFirst.php
```

You should have a result like this :

```
PHPUnit 7.3.5 by Sebastian Bergmann and contributors.

.                                                                   1 / 1 (100%)

Time: 2.01 seconds, Memory: 10.00MB

There was 1 failure:
```

(continues on next page)

(continued from previous page)

```

1) Test\Custom_MyFirst::testCustom_MyFirst01
1 values were found and are unprocessed : 0 + $a

source/Custom/MyFirst.02.php
exp/Custom/MyFirst.02.php
phpunit --filter=01 Test/Custom/MyFirst.php


Failed asserting that 1 matches expected 0.

FAILURES!
Tests: 2, Assertions: 5, Failures: 1.
```

In the first analysis, we have build a query to look for *$a + 0* but not for *0 + $a*. It is a good thing that we added tests for them, so we need to add more query to the analysis.

Open again the 'library/Exakat/Analyze/Custom/MyFirst.php', and, inside the *analyze()* method, below the first *prepareQuery()*, add the following code to search for *0 + $a* :

```
$this->atomIs('Addition')
     ->codeIs('+')
     ->outIs('LEFT')
     ->atomIs('Integer')
     ->codeIs('0');
$this->prepareQuery();
```

An analysis may run several queries. In this case, we have searched for '$a + 0', but we should also check for '0 + $a'. Addition is associative, so 0 may be useless on the right or on the left.

```
PHPUnit 7.3.5 by Sebastian Bergmann and contributors.

.                                                                   1 / 1 (100%)

Time: 2.82 seconds, Memory: 10.00MB
```

This means that the Unit Test found the values we expected, and it also didn't find the values we didn't want.

Congratulations! This is your first analysis, and it is time to celebrate! Welcome to the great family of static analyzers.

13.4 Analysis structure

An analysis class is build with 4 elements.

- analyze() method
- dependsOn() method
- $phpVersion property
- Analyzer extends

13.4.1 The analyze() method

The analyze method is the most important. It is the method that does the actual analysis.

The method doesn't return anything.

13.4.2 The dependsOn() method

This method returns the list of other analysis on which the current analysis depends on. For example, an analysis may target PHP functions : it relies on another analysis that detects the PHP functions, then, add it own layer of review. The other analysis must be processed first, and the Exakat engine run the dependencies before it runs the current analysis.

The list of dependency is a array of strings, with the usual analysis format : for example, *array('Functions/IsExtFunction')*. Multiple analysis may be returned by that method. If the current analysis is autonomous, the method may be omitted, or it may return an empty array.

13.4.3 $phpVersion property

The protected *$phpVersion* property configure the analysis to run with specific versions of PHP. For example, *Structures/Break0* is an analysis that can only run until PHP 5.4 : after that version, PHP doesn't compile code that uses *break 0;*. Thus, there is no need to run the current analysis on newer PHP versions.

If the analysis works on every PHP version, this can be omitted.

Patch level version are never taken into consideration : PHP 7.0.0 or PHP 7.0.30 are all covered by '7.0'.

$phpVersion accepts several values :

- '7.0' : the analysis only runs for PHP 7.0 version.
- '7.1-' : the analysis only runs until PHP 7.1 version. PHP 7.1 is excluded.
- '7.2+' : the analysis only runs after PHP 7.2 version. PHP 7.2 is included.

Generally speaking, PHP version are the official middle versions : 5.2, 5.3, 5.4, 5.5, 5.6, 7.0, 7.1, 7.2, 7.3, 7.4, 8.0. This changes with the publication of PHP versions.

13.4.4 Analyzer extends

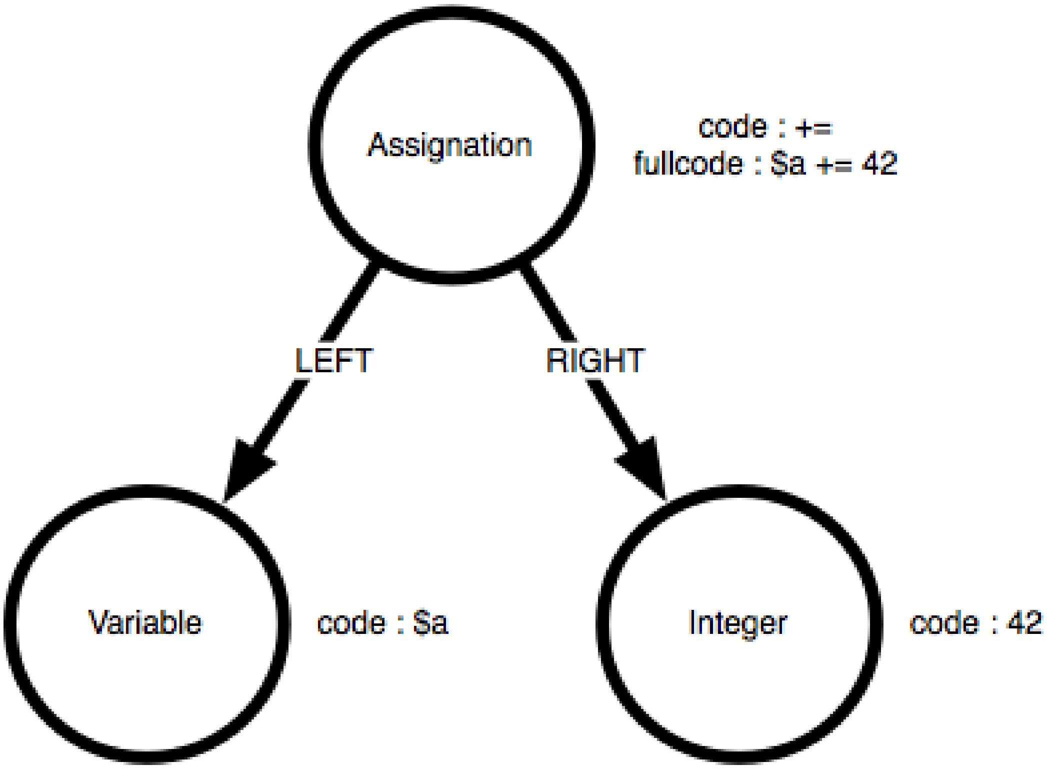
By default, an analysis extends the ExakatAnalyzerAnalyzer class. Some frequent analysis that can be configurer, are available in the *Common/** folder. More on that later.

13.5 Internal database

13.5.1 Presentation

Every important structure of PHP code is stored in the database as a node, called atom. Nodes are connected to each other with links. Each atom has a list of defining properties, that are not represented in the code. For example, where is a simple assignation :

The 'Assignation' atom is holding the '+=' code, which is its characteristics. Then, it has two members : 'LEFT' and 'RIGHT'. Each of the target atoms are different : one is a variable, and the other is a integer. Altogether, they build the assignation, which is summed up in the 'fullcode' property of the assignation.



To define a pattern in the code, we use a combinaison of filters on atom, links or their property. Any succession of steps that yield a result means that an issue has been found in the code.

13.5.2 Atoms

Here is the list of the 117 available atoms :

- Addition : An addition or a substraction
- Analysis : An analysis, as processed by Exakat.
- Array : Represents array access : *$a[4]*, *$this['a']['f']* and *foo()[1]['g']*
- Arrayappend : Represents *$a[]* or *$this->b[]*
- Arrayliteral : Represents an array definition : *[4,5 => 3,6]* and *array(1,2,3)*
- As : The as keyword, when aliasing an imported class
- Assignation : Any assignation and short assignation : *$a = 1*, *$b .= 3*
- Bitshift : A bit shift operation on integers, with << or >>
- Block : Represents a sequence between curly braces. For example, *{ $c += $b; }*.
- Boolean : Represents *true* or *false*.
- Break : A break, with or without the level indication. *break 1*;
- Cast : A case expression in a switch() statement. 'case 1: '
- Cast : A cast operation, like `(array)` or `(unset)`
- Catch : A catch clause in a try/catch command. For example : *catch (Exception $e)* or *catch{A|B|C $d}*

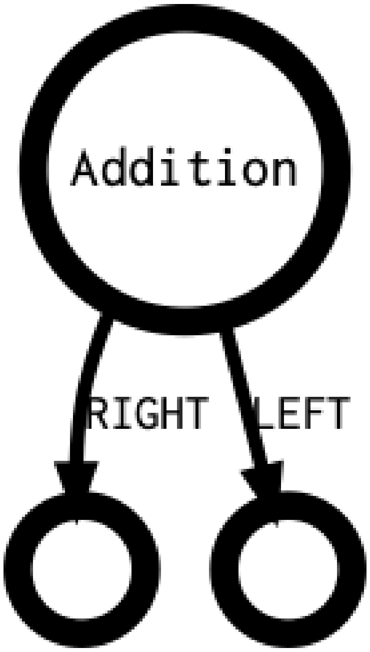
- Class : A named class.
- Classalias : A call to the *class_alias* function.
- Classanonymous : A unnamed class, created with *new class {};*
- Clone : A clone expression
- Closure : A closure definition. For example, *function () { return 3; }*.
- Coalesce : An expression with the coalesce operator, *?:*. For example, *$x = $y ?: 'ef';*
- Comparison : A comparison, with any kind of comparison operator : ==, ===, >, ...
- Concatenation : A concatenation : a sequence of values, linked by the dot operator .
- Const : A constant definition, for classes or global. *const X = 1;* or *class x { const Y = 2; }*
- Constant : A constant definition, part of a *Const* atom.
- Continue : A continue operator, with or without its level indicator
- Declare : A declare expression.
- Declaredefinition : One configuration expression inside a *declare* definition. For example, in *declare(strict_types=1);*, *strict_types=1*
- Default : A default case, in a switch statement.
- Defineconstant : A call to the *define()* function.
- Dowhile : A do... while() loop.
- Echo : A call to *echo*
- Empty : A call to *empty*
- Eval : A call to *Eval*
- Exit : A call to *Exit*
- File : A file, containing the PHP source code.
- Finally : A finally clause in a try/catch command.
- For : A for loop. For example : *for($i = 0; $i < 10; ++$i) { }*
- Foreach : A foreach loop.
- Function : A function definition
- Functioncall : A call to a function.
- Global : An expression with the global keyword. For example, *global $x, $y*.
- Globaldefinition : A definition of a global variable, inside a global expression. For example, in *global $x = 1, $y*, *$x = 1* and *$y* are Globaldefinition.
- Goto : The goto expression.
- Gotolabel : A target destination for a goto expression.
- Halt : The *__halt_compiler* command.
- Heredoc : A Heredoc or Nowdoc string
- Identifier : A name for a constant or a class. For example : *$x instanceof Y*, 'echo PHP_INT_MAX', *new Y*
- Ifthen : A if/then/else structure.
- Include : A inclusion, with *require* or *include*, with *_once* or not.

- Inlinehtml : Raw text, in the middle of a PHP script. For example : ‘‘++$a; ?>RAW TEXT<?php ++$b; ‘‘
- Instanceof : A *instanceof* expression
- Insteadof : A *insteadof* expression
- Integer : An Integer literal, positive or negative.
- Interface : An interface definition
- Isset : A call to *isset*
- Keyvalue : An expression with the => operator : for arrays or foreach() instructions.
- List : The list() or [] call when on the right of an assignation.
- Logical : A logical expression. This covers also bitwise operations. For example : *$a | $b*, *$a && $b*, *$a xor $b*.
- Magicconstant : A PHP magic constant. For example : __FILE__ or __class__.
- Magicmethod : A special PHP method in a class. For example, *__clone()*, *__construct()*, *__get()*, . . .
- Member : A reference to a member of an object. For example, *$object->member*.
- Method : A method definition in a class.
- Methodcall : A non-static call to a method. For example, *$a->method();*
- Methodcallname : The name of the method in a methodcall
- Multiplication : A multiplication *, division / or modulo % operation.
- Name : The name of a structure : name of a class, method, interface, trait, interface.
- Namespace : A namespace declaration
- New : An instantiation expression, with *new ClassName()*.
- Newcall : The functioncall in a New expression. For example, in ‘‘new foo()‘, *foo()* is the Newcall.
- Not : A call to *!* or ~.
- Nsname : A fully qualified name, including ‘*. For example, ‘strtolower*, *ABC*, . . . w
- Null : The *Null* value
- Parameter : A parameter definition, in a function or method definition. When called, it becomes an argument.
- Parametername : A Parametername
- Parent : The parent keyword, when it is used to refer to the parent class.
- Parenthesis : A Parenthesis expression. This is not a syntactic parenthesis, like in a switch or functioncall.
- Php : A PHP script, inside its tags. This exclude the following and previous raw text in a PHP file.
- Phpvariable : A PHP reserved variable, such as *$_GET*, *$_POST*, *$GLOBALS*, etc.
- Postplusplus : $i++‘ expression
- Power : The power operator, **.
- Ppp : A properties declaration, in a class or a trait. For example : *private $x, $y = 2;*
- Preplusplus : ++ or – when it is before the variable.
- Print : A call to the function print.
- Project : The project node : the root above all File.

- Propertydefinition : A property definition. For example : `class x { private $property = 1; var $x; }`
- Real : A float number
- Return : The return expression.
- Self : The *self* keyword, as used inside a class.
- Sequence : A virtual atom, that represents the sequence of expression, in a block.
- Shell : A shell, made with ticks ‘
- Sign : A Sign structure : when a -‘*or* ‘+ has been added before another expression. For example *- ($a + $b)*.
- Static : The static keyword, when it is used to refer to the current class.
- Staticclass : A call to *::class*, with the syntax of a static constant. For example, *X::class*.
- Staticconstant : A staticconstant : *TheClass::TheConstant*
- Staticdefinition : A static variable definition, in a method or function. This is not a static property. For example ; *function foo() { static $s; }*.
- Staticmethod : A staticmethod name, when using trait and renaming a method. For example, *trait t { use t2 { C::D as E; }}*. *C::D* is a static method.
- Staticmethodcall : A static methodcall
- Staticproperty : A static property syntax. For example, *A::$b* or *self::$d*.
- Staticpropertyname : The name of a static property : not a variable.
- String : A string literal, with or without interpolation. For example, *‘$x’*, *“a{$y}”*, *“a”*.
- Switch : A switch structure.
- Ternary : The ternary operator : *$a ? $b : ‘c’*.
- This : The special variable *$this*.
- Throw : A throw expression
- Trait : A trait. For example : *trait t { function foo() {} }*
- Try : The Try part in a try/catch/finally expression.
- Unset : A call to *unset*
- Usenamespace : Use expression within a namespace, and not in a class or trait.
- Usetrait : A *use* expression, when used to import a trait. For exapmle, *class x { use t; }*
- Variable : A Variable, as a standalone container. For example : *$a = 1* or *$b += 3*. Variables in arrays are *Variablearray*, while variables in objects are *Variableobject*.
- Variablearray : A variable, when used to build an array syntax. For example, the *$x* in *$x[0]* or *$x[]*.
- Variabledefinition : A placeholder to federate local variable definition in a method.
- Variableobject : A variable when used with the -> operator.
- Void : A Void operation. It represents the absence of data. For example : *foo();;* : there is a Void as argument, and one between the semicolons.
- While : A While structure, different from a Dowhile structure. For example : *while($a < 10) { $a++;}*
- Yield : A *yield* expression
- Yieldfrom : A *yield from* expression

Addition

An addition or a substraction



List of available properties :

- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- intval
- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- T_MINUS
- T_PLUS

List of outgoing links :

- LEFT
- RIGHT

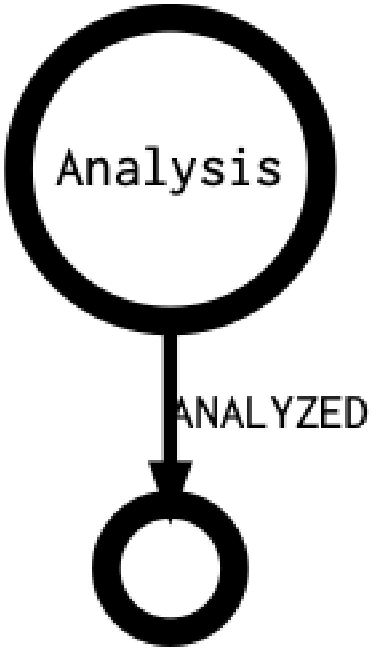
List of incoming links :

- ANALYZED

- ARGUMENT
- CODE
- CONCAT
- CONDITION
- DEFAULT
- ELSE
- EXPRESSION
- INDEX
- LEFT
- RETURN
- RIGHT
- SOURCE
- THEN
- VALUE

Analysis

An analysis, as processed by Exakat.



List of available properties :

- Atom
- analyzer
- atom

List of possible tokens :

-

List of outgoing links :

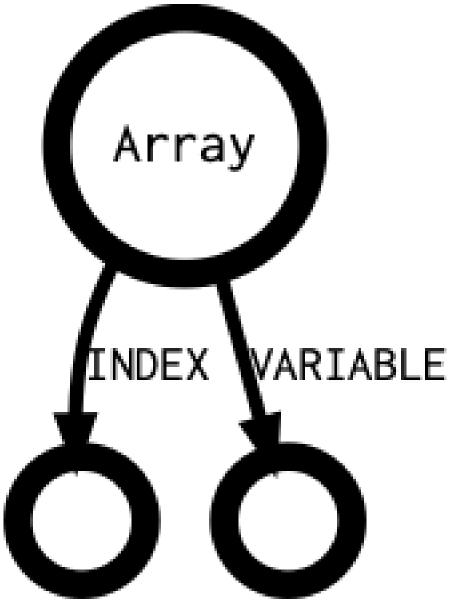
- ANALYZED

List of incoming links :

-

Array

Represents array access : *$a[4]*, *$this['a']['f']* and *foo()[1]['g']*



List of available properties :

- code
- ctype1
- ctype1_size
- enclosing
- fullcode
- globalvar
- lccode
- line
- noDelimiter
- noscream
- rank
- reference
- variadic

List of possible tokens :

- T_CLOSE_BRACKET
- T_CLOSE_CURLY
- T_CLOSE_PARENTHESIS
- T_CONSTANT_ENCAPSED_STRING
- T_CURLY_OPEN
- T_QUOTE
- T_START_HEREDOC
- T_STRING
- T_STRING_VARNAME
- T_VARIABLE

List of outgoing links :

- INDEX
- VARIABLE

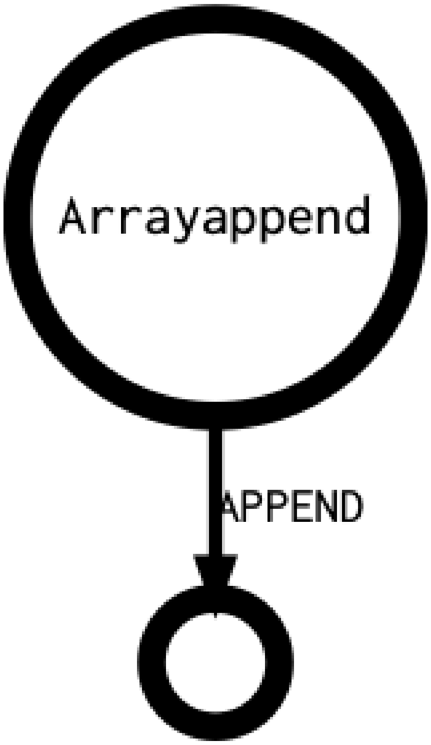
List of incoming links :

- ANALYZED
- APPEND
- ARGUMENT
- CASE
- CAST
- CLASS
- CLONE
- CODE
- CONCAT
- CONDITION
- DEFAULT
- ELSE
- EXPRESSION
- INDEX
- LEFT
- NAME
- NEW
- NOT
- OBJECT
- POSTPLUSPLUS
- PREPLUSPLUS
- RETURN

- RIGHT
- SIGN
- SOURCE
- THEN
- THROW
- VALUE
- VARIABLE

Arrayappend

Represents *$a[]* or *$this->b[]*



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank
- reference

List of possible tokens :

- T_CLOSE_BRACKET
- T_CLOSE_CURLY
- T_CLOSE_PARENTHESIS
- T_STRING
- T_VARIABLE

List of outgoing links :

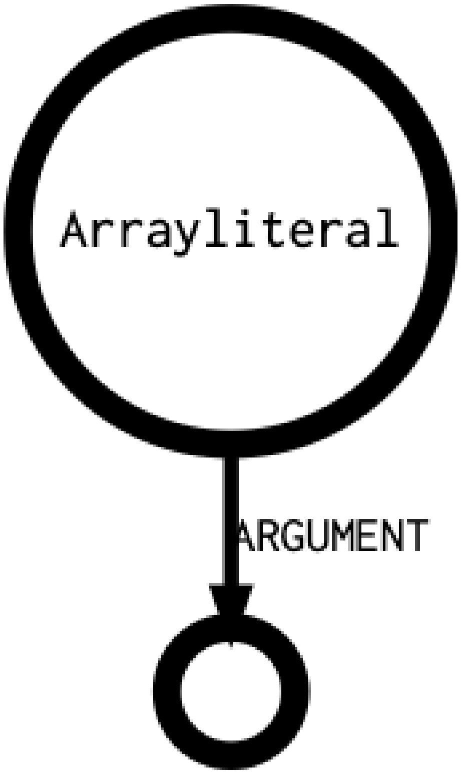
- APPEND

List of incoming links :

- APPEND
- ARGUMENT
- LEFT
- OBJECT
- POSTPLUSPLUS
- PREPLUSPLUS
- RETURN
- RIGHT
- VALUE
- VARIABLE
- YIELD

Arrayliteral

Represents an array definition : *[4,5 => 3,6]* and *array(1,2,3)*



List of available properties :

- args_max
- args_min
- boolean
- cbClass
- cbMethod
- cbObject
- code
- constant
- count
- ctype1
- ctype1_size
- fullcode
- intval
- lccode
- line
- noDelimiter
- noscream
- rank
- variadic

List of possible tokens :

- T_ARRAY
- T_OPEN_BRACKET

List of outgoing links :

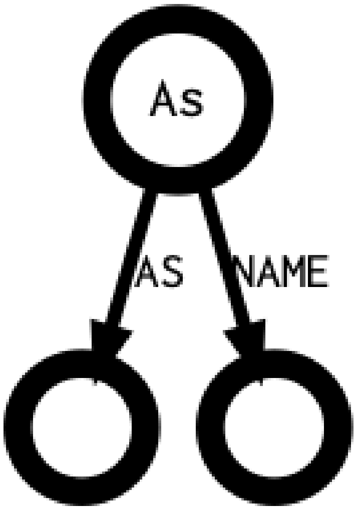
- ARGUMENT

List of incoming links :

- ANALYZED
- ARGUMENT
- CASE
- CAST
- CLONE
- CODE
- DEFAULT
- DEFINITION
- ELSE
- EXPRESSION
- INDEX
- LEFT
- NAME
- OBJECT
- RETURN
- RIGHT
- SOURCE
- THEN
- VALUE
- VARIABLE
- YIELD

As

The as keyword, when aliasing an imported class



List of available properties :

- alias
- code
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- rank
- visibility

List of possible tokens :

- T_AS

List of outgoing links :

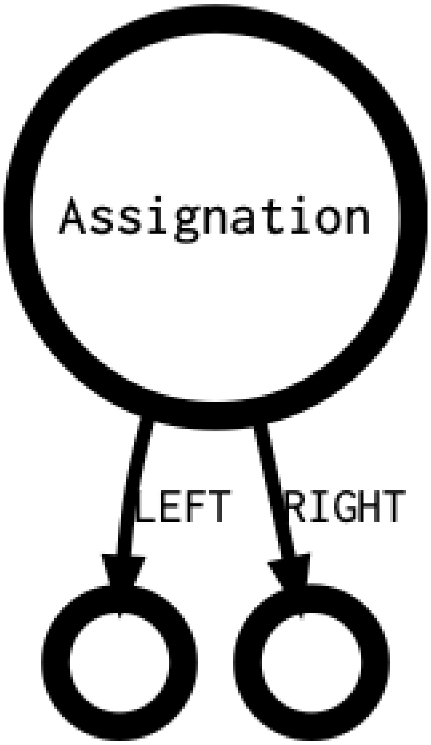
- AS
- DEFINITION
- NAME

List of incoming links :

- ANALYZED
- DEFINITION
- EXPRESSION
- USE

Assignation

Any assignation and short assignation : *$a = 1*, *$b .= 3*



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_AND_EQUAL
- T_CONCAT_EQUAL
- T_DIV_EQUAL
- T_EQUAL
- T_MINUS_EQUAL
- T_MOD_EQUAL
- T_MUL_EQUAL
- T_OR_EQUAL
- T_PLUS_EQUAL
- T_POW_EQUAL
- T_SL_EQUAL

- T_SR_EQUAL
- T_XOR_EQUAL

List of outgoing links :

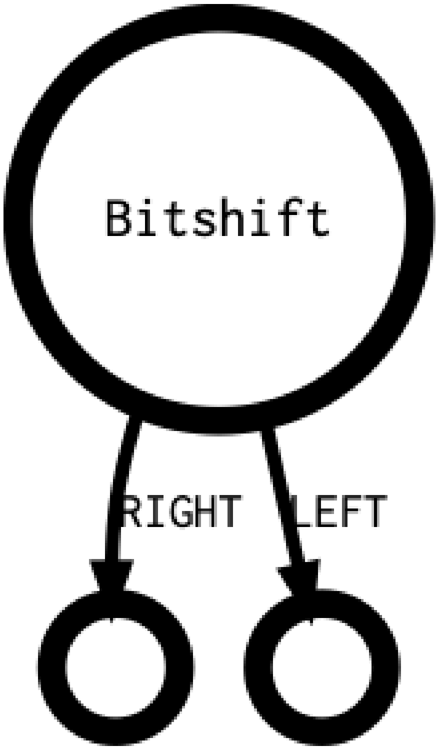
- LEFT
- RIGHT

List of incoming links :

- ANALYZED
- ARGUMENT
- CODE
- CONDITION
- ELSE
- EXPRESSION
- INDEX
- LEFT
- NAME
- RETURN
- RIGHT
- SOURCE
- THEN
- THROW
- VALUE

Bitshift

A bit shift operation on integers, with << or >>



List of available properties :

- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- intval
- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- T_SL
- T_SR

List of outgoing links :

- LEFT
- RIGHT

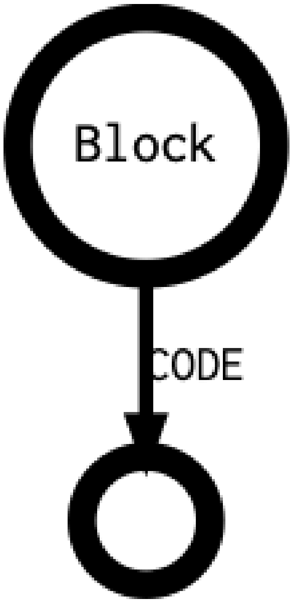
List of incoming links :

- ANALYZED

- ARGUMENT
- CODE
- CONDITION
- DEFAULT
- LEFT
- RETURN
- RIGHT
- VALUE

Block

Represents a sequence between curly braces. For example, *{ $c += $b; }*.



List of available properties :

- code
- ctype1_size
- fullcode
- lccode
- line

List of possible tokens :

- T_CLOSE_BRACKET
- T_CLOSE_CURLY
- T_CLOSE_PARENTHESIS
- T_CONSTANT_ENCAPSED_STRING
- T_LNUMBER

- T_QUOTE
- T_STRING
- T_VARIABLE

List of outgoing links :

- CODE

List of incoming links :

- MEMBER
- NAME

Boolean

Represents *true* or *false*.

List of available properties :

- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- fullnspath
- intval
- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- T_STRING

List of outgoing links :

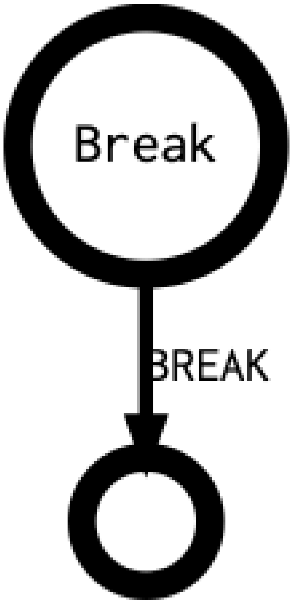
-

List of incoming links :

- ANALYZED
- ARGUMENT
- CASE
- CODE
- CONCAT
- CONDITION
- DEFAULT
- ELSE
- EXPRESSION
- INDEX
- LEFT
- RETURN
- RIGHT
- THEN
- VALUE

Break

A break, with or without the level indication. *break 1*;



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode

- lccode
- line
- rank

List of possible tokens :

- T_BREAK

List of outgoing links :

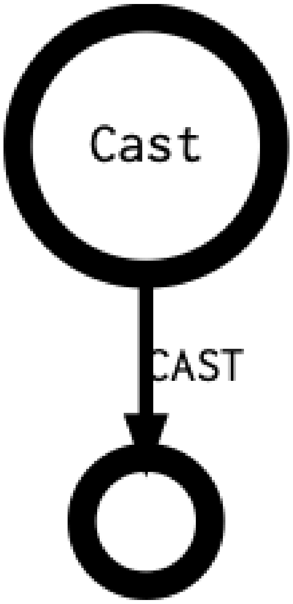
- BREAK

List of incoming links :

- EXPRESSION

Cast

A case expression in a switch() statement. ‘case 1: ‘



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_CASE

List of outgoing links :

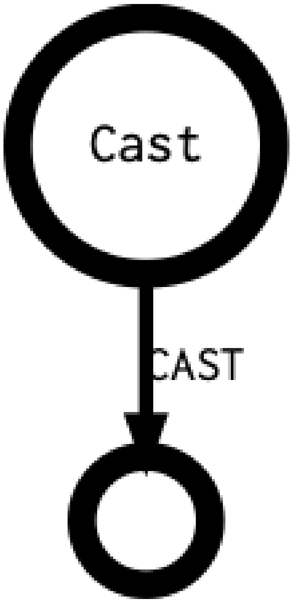
- CASE
- CODE

List of incoming links :

- ANALYZED
- EXPRESSION

Cast

A cast operation, like `(array)` or `(unset)`



List of available properties :

- binaryString
- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_ARRAY_CAST
- T_BOOL_CAST
- T_DOUBLE_CAST
- T_INT_CAST
- T_OBJECT_CAST

- T_STRING_CAST

List of outgoing links :

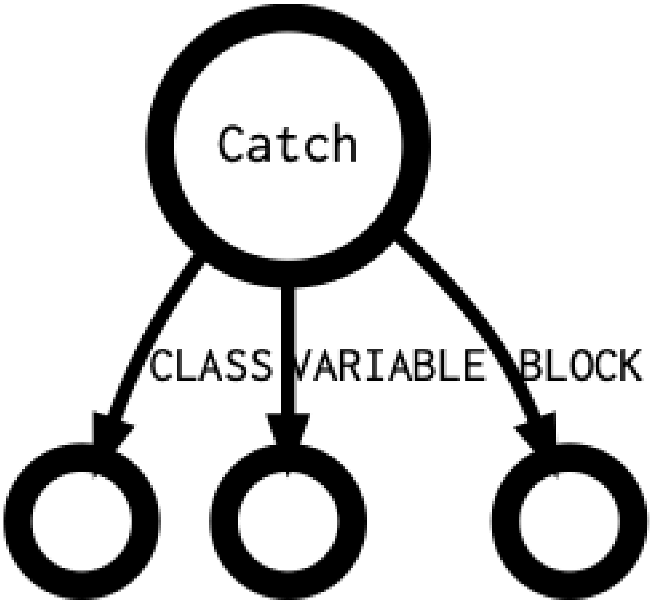
- CAST

List of incoming links :

- ANALYZED
- ARGUMENT
- CAST
- CODE
- CONCAT
- CONDITION
- ELSE
- EXPRESSION
- INDEX
- LEFT
- NAME
- POSTPLUSPLUS
- RETURN
- RIGHT
- SIGN
- SOURCE
- THEN
- VALUE
- YIELD

Catch

A catch clause in a try/catch command. For example : *catch (Exception $e)* or *catch{A|B|C $d}*



List of available properties :

- code
- count
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_TRY
- 1

List of outgoing links :

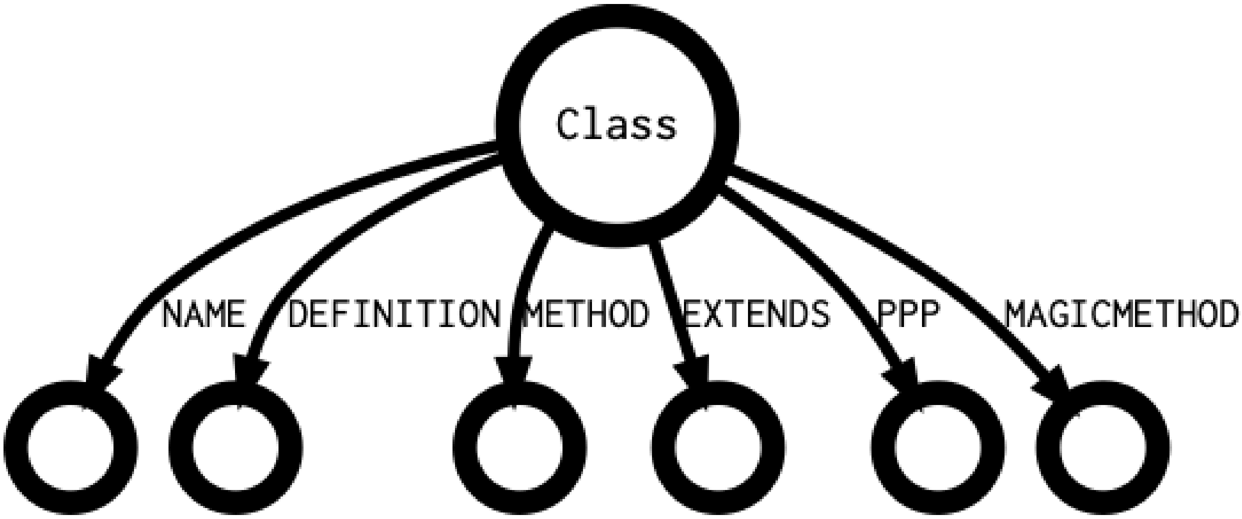
- BLOCK
- CLASS
- VARIABLE

List of incoming links :

- ANALYZED
- CATCH

Class

A named class.



List of available properties :

- abstract
- aliased
- code
- ctype1_size
- final
- fullcode
- fullnspath
- lccode
- line
- rank

List of possible tokens :

- T_CLASS
- 1

List of outgoing links :

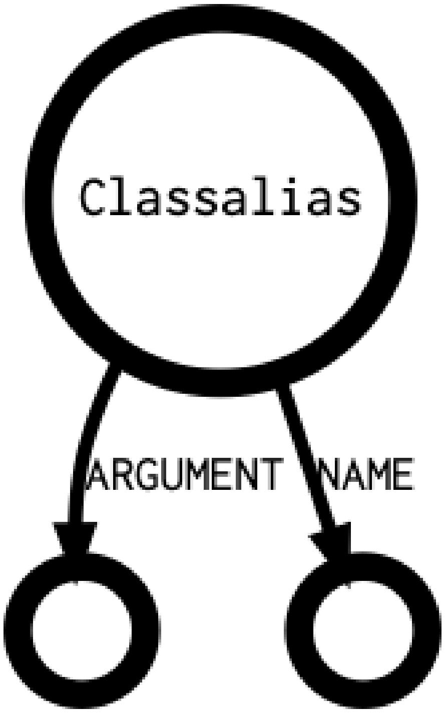
- CONST
- DEFINITION
- EXTENDS
- IMPLEMENTS
- MAGICMETHOD
- METHOD
- NAME
- PPP
- USE

List of incoming links :

- ANALYZED
- EXPRESSION

Classalias

A call to the *class_alias* function.



List of available properties :

- args_max
- args_min
- code
- constant
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- rank

List of possible tokens :

- T_STRING

List of outgoing links :

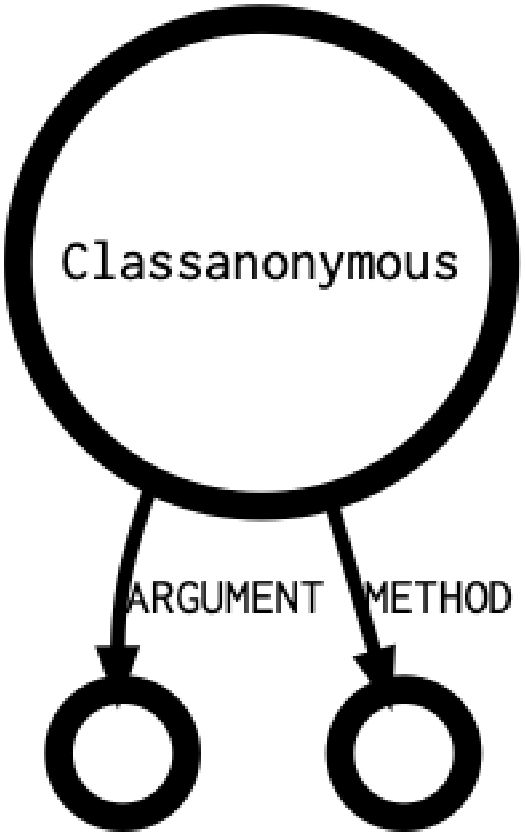
- ARGUMENT
- NAME

List of incoming links :

- ANALYZED
- EXPRESSION
- RIGHT

Classanonymous

A unnamed class, created with *new class {};*



List of available properties :

- args_max
- args_min
- code
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath

- lccode
- line

List of possible tokens :

- T_CLASS

List of outgoing links :

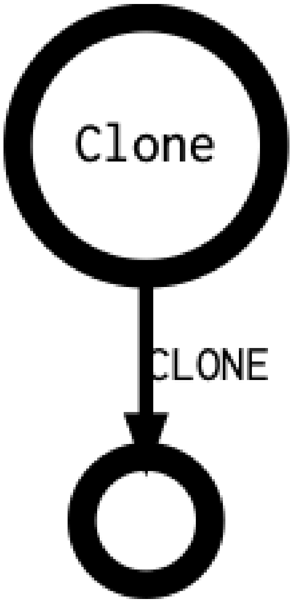
- ARGUMENT
- DEFINITION
- EXTENDS
- IMPLEMENTS
- MAGICMETHOD
- METHOD
- PPP
- USE

List of incoming links :

- ANALYZED
- NEW

Clone

A clone expression



List of available properties :

- code
- ctype1
- ctype1_size

- fullcode
- lccode
- line
- noscream
- rank

List of possible tokens :

- T_CLONE

List of outgoing links :

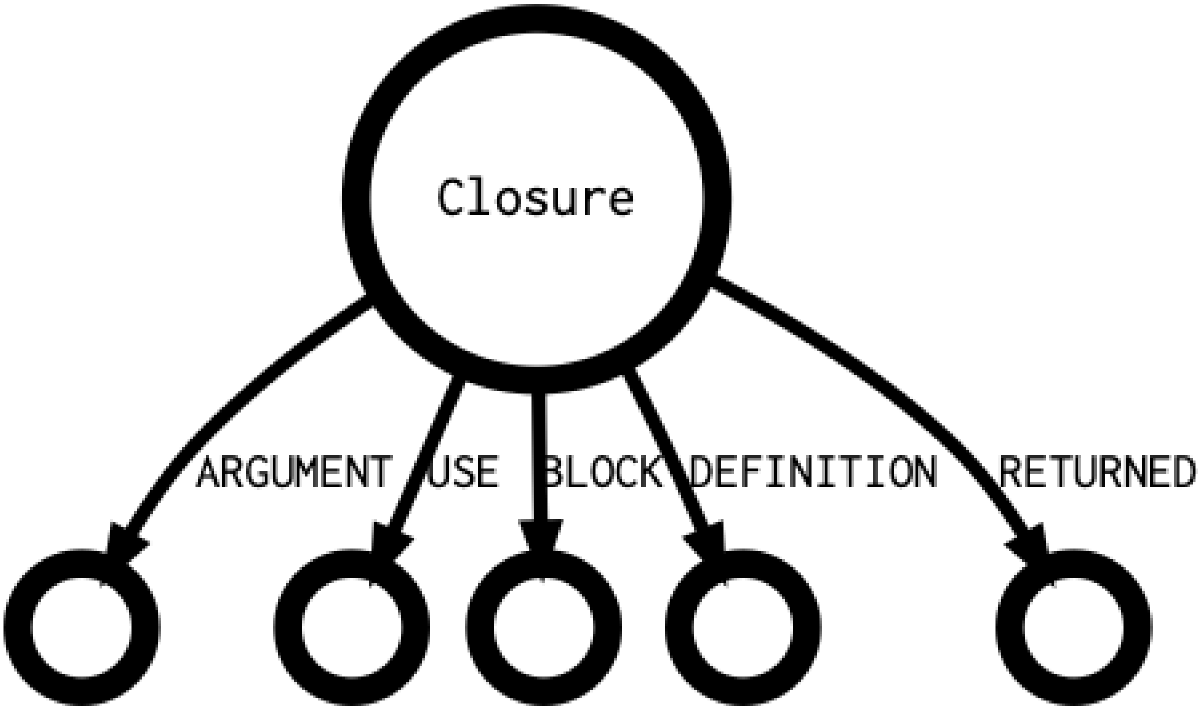
- CLONE

List of incoming links :

- ANALYZED
- ARGUMENT
- CLONE
- CODE
- ELSE
- EXPRESSION
- LEFT
- RETURN
- RIGHT
- SOURCE
- THEN

Closure

A closure definition. For example, *function () { return 3; }*.



List of available properties :

- args_max
- args_min
- boolean
- code
- constant
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- noscream
- nullable
- rank
- reference
- static

List of possible tokens :

- T_FUNCTION

List of outgoing links :

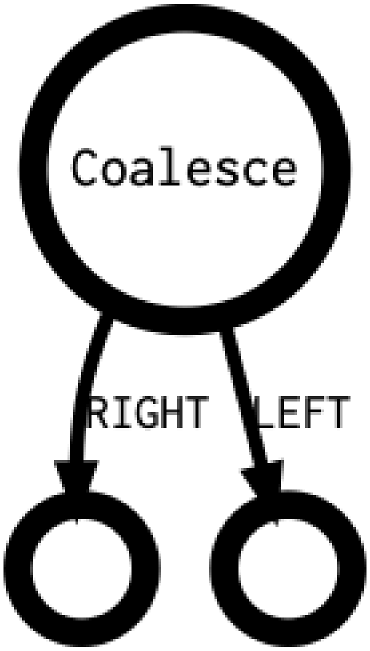
- ARGUMENT
- BLOCK
- DEFINITION
- RETURNED
- RETURNTYPE
- USE

List of incoming links :

- ANALYZED
- ARGUMENT
- CAST
- CODE
- ELSE
- INDEX
- RETURN
- RIGHT
- SOURCE
- THEN
- VALUE
- VARIABLE

Coalesce

An expression with the coalesce operator, *?:*. For example, *$x = $y ?: 'ef';*



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- intval
- isNull
- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- T_COALESCE

List of outgoing links :

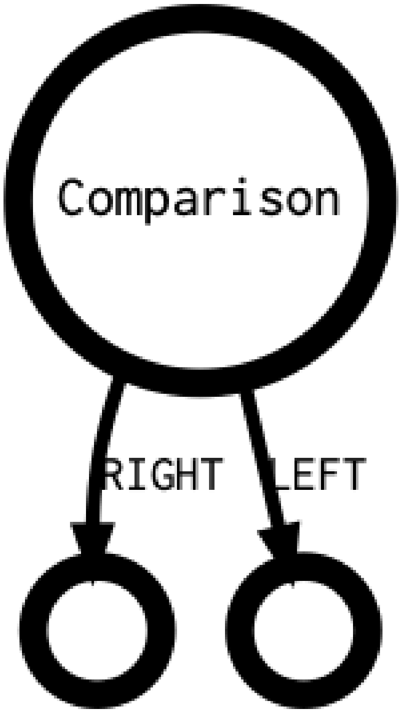
- LEFT
- RIGHT

List of incoming links :

- ARGUMENT
- CODE
- CONDITION
- DEFAULT
- RETURN
- RIGHT
- SOURCE
- THEN
- VALUE

Comparison

A comparison, with any kind of comparison operator : ==, ===, >, . . .



List of available properties :

- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- intval
- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- T_GREATER
- T_IS_EQUAL
- T_IS_GREATER_OR_EQUAL
- T_IS_IDENTICAL
- T_IS_NOT_EQUAL
- T_IS_NOT_IDENTICAL
- T_IS_SMALLER_OR_EQUAL
- T_SMALLER

- T_SPACESHIP

List of outgoing links :

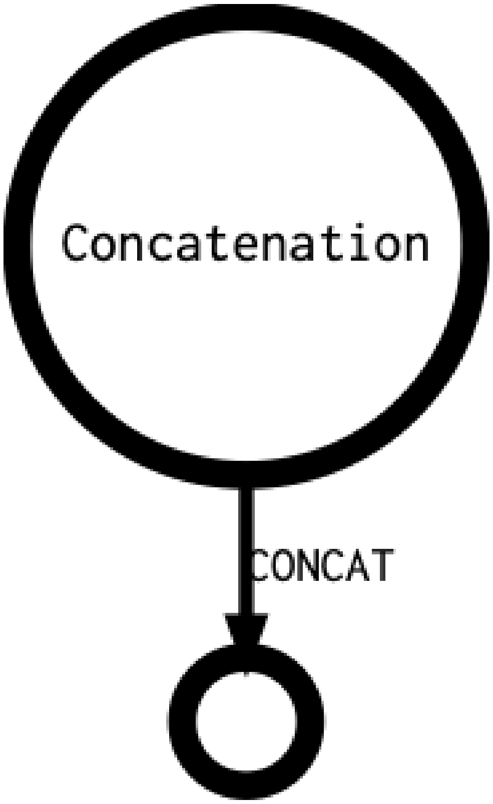
- LEFT
- RIGHT

List of incoming links :

- ANALYZED
- ARGUMENT
- CASE
- CODE
- CONDITION
- ELSE
- EXPRESSION
- INDEX
- LEFT
- RETURN
- RIGHT
- THEN
- VALUE

Concatenation

A concatenation : a sequence of values, linked by the dot operator .



List of available properties :

- boolean
- code
- constant
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- intval
- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- T_DOT

List of outgoing links :

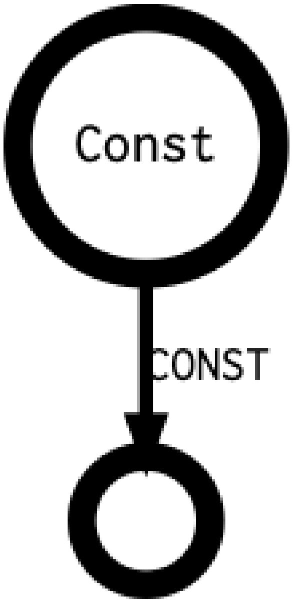
- CONCAT

List of incoming links :

- ANALYZED
- ARGUMENT
- CASE
- CODE
- CONDITION
- DEFAULT
- ELSE
- EXPRESSION
- INDEX
- LEFT
- NAME
- RETURN
- RIGHT
- THEN
- THROW
- VALUE

Const

A constant definition, for classes or global. *const X = 1;* or *class x { const Y = 2; }*



List of available properties :

- code
- count
- ctype1_size

- fullcode
- lccode
- line
- rank
- visibility

List of possible tokens :

- T_CONST
- 1

List of outgoing links :

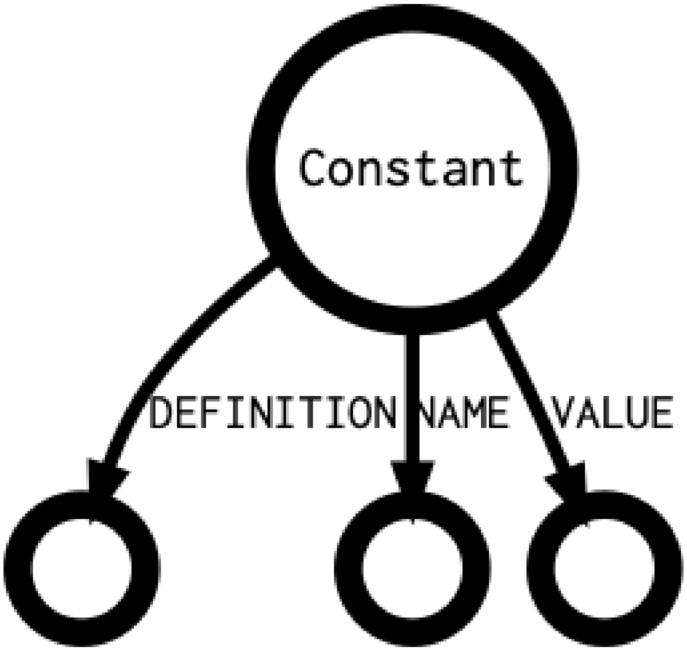
- CONST

List of incoming links :

- ANALYZED
- CONST
- EXPRESSION

Constant

A constant definition, part of a *Const* atom.



List of available properties :

- boolean
- code
- ctype1
- ctype1_size

- fullcode
- intval
- isNull
- lccode
- line
- rank

List of possible tokens :

- T_COMMA
- T_CONST
- 1

List of outgoing links :

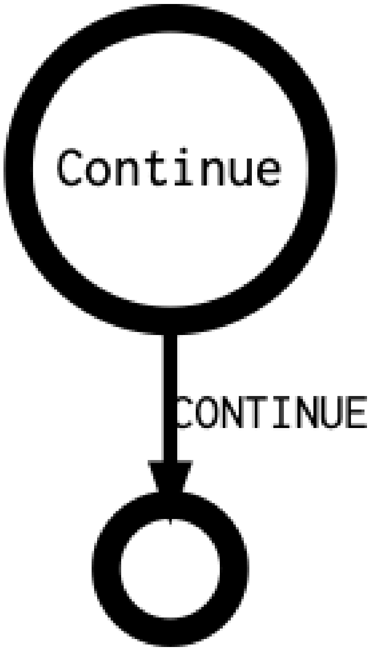
- DEFINITION
- NAME
- OVERWRITE
- VALUE

List of incoming links :

- ANALYZED
- CONST
- OVERWRITE

Continue

A continue operator, with or without its level indicator



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_CONTINUE

List of outgoing links :

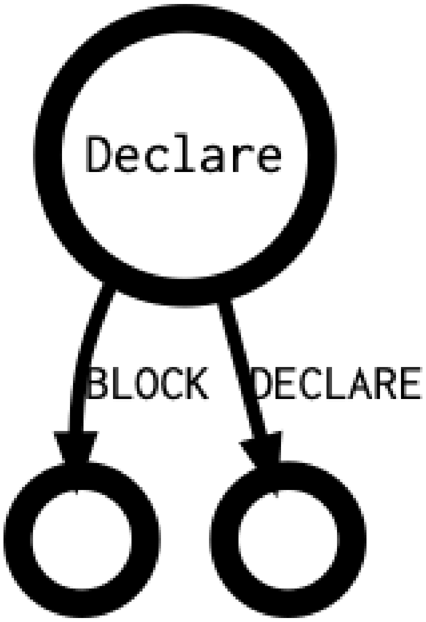
- CONTINUE

List of incoming links :

- ANALYZED
- EXPRESSION

Declare

A declare expression.



List of available properties :

- code
- ctype1_size
- fullcode
- lccode

- line
- rank

List of possible tokens :

- T_DECLARE

List of outgoing links :

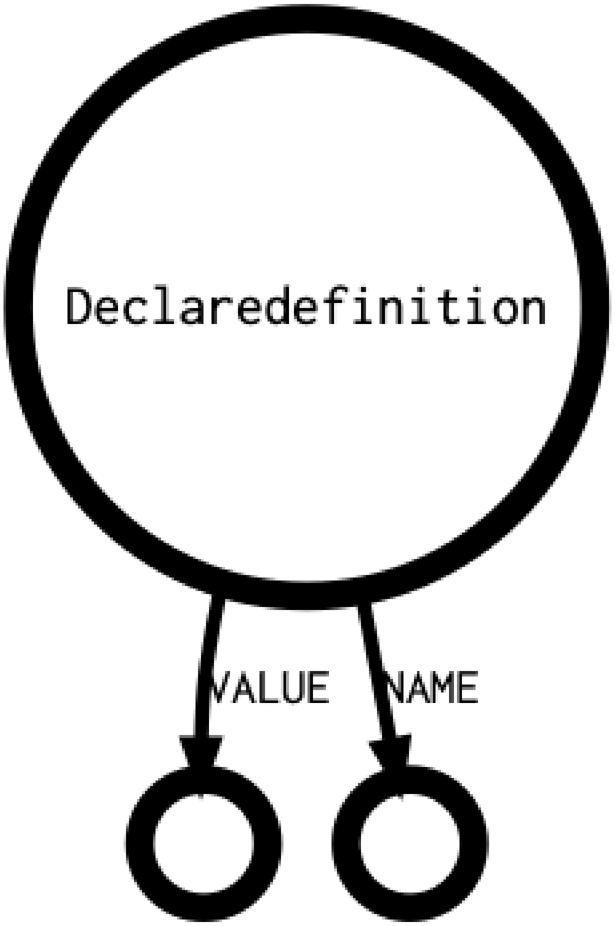
- BLOCK
- DECLARE

List of incoming links :

- ANALYZED
- EXPRESSION

Declaredefinition

One configuration expression inside a *declare* definition. For example, in *declare(strict_types=1);*, *strict_types=1*



List of available properties :

- code
- ctype1_size

- fullcode
- lccode
- line

List of possible tokens :

-

List of outgoing links :

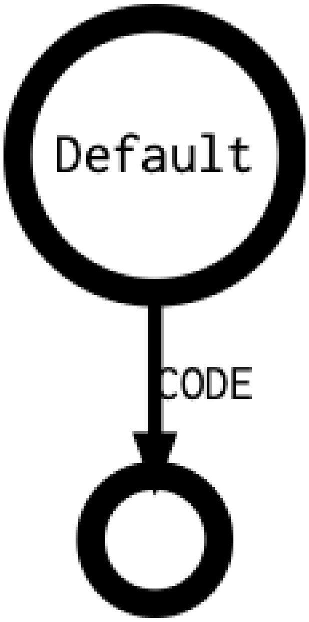
- NAME
- VALUE

List of incoming links :

- DECLARE

Default

A default case, in a switch statement.



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_DEFAULT

List of outgoing links :

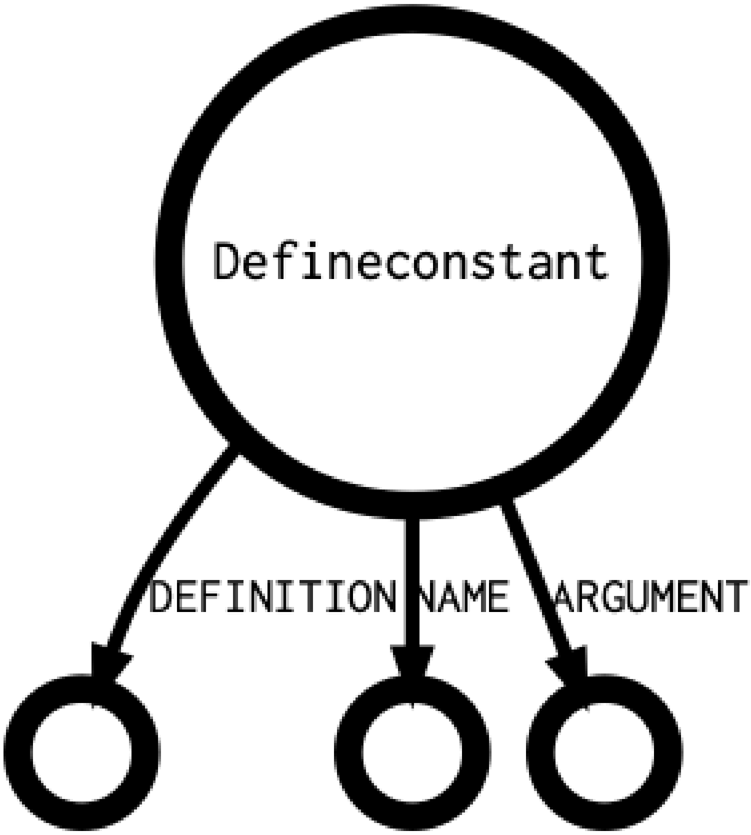
- CODE

List of incoming links :

- EXPRESSION

Defineconstant

A call to the *define()* function.



List of available properties :

- args_max
- args_min
- code
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line

- rank

List of possible tokens :

- T_STRING

List of outgoing links :

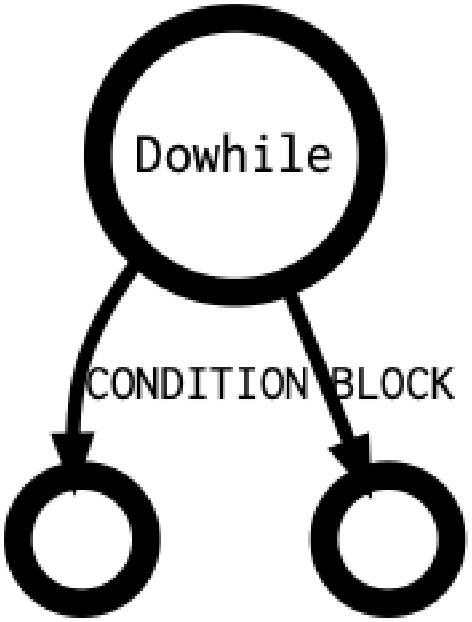
- ARGUMENT
- DEFINITION
- NAME

List of incoming links :

- ANALYZED
- ARGUMENT
- EXPRESSION
- RIGHT

Dowhile

A do. . . while() loop.



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_DO

List of outgoing links :

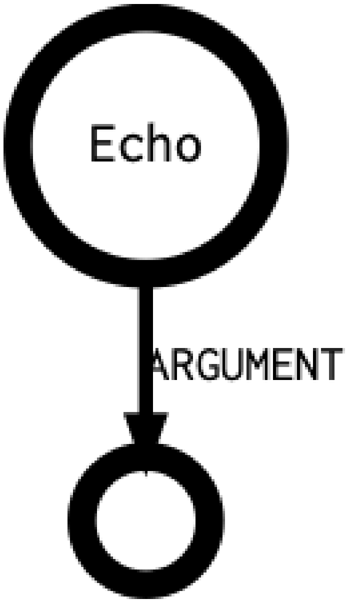
- BLOCK
- CONDITION

List of incoming links :

- ANALYZED
- EXPRESSION

Echo

A call to *echo*



List of available properties :

- args_max
- args_min
- code
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- rank

List of possible tokens :

- T_ECHO
- T_OPEN_TAG_WITH_ECHO
- 1

List of outgoing links :

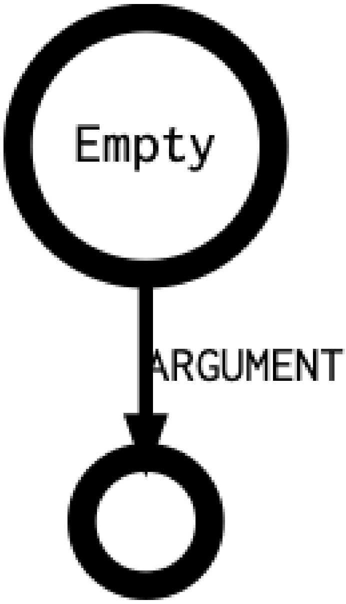
- ARGUMENT
- NAME

List of incoming links :

- ANALYZED
- EXPRESSION

Empty

A call to *empty*



List of available properties :

- args_max
- args_min
- code
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode

- line
- rank

List of possible tokens :

- T_EMPTY

List of outgoing links :

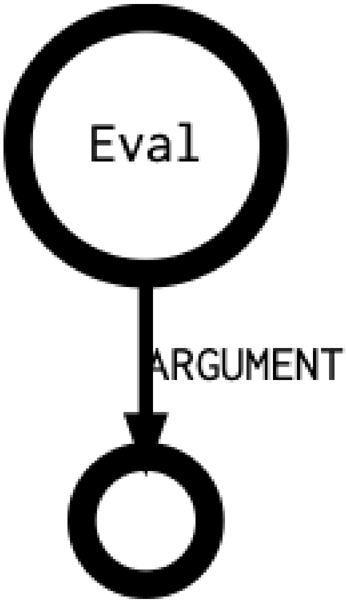
- ARGUMENT

List of incoming links :

- ANALYZED
- ARGUMENT
- CODE
- CONDITION
- LEFT
- NOT
- RETURN
- RIGHT
- VALUE

Eval

A call to *Eval*



List of available properties :

- args_max
- args_min
- code

- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- noscream
- rank

List of possible tokens :

- T_EVAL

List of outgoing links :

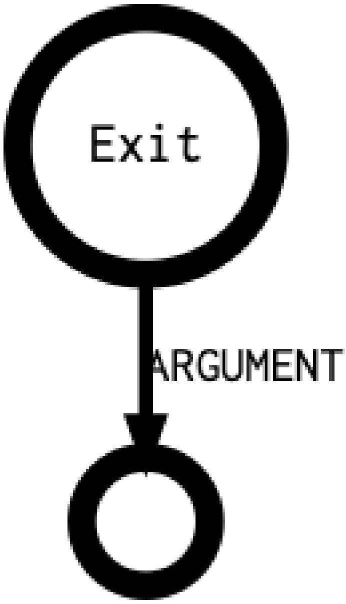
- ARGUMENT

List of incoming links :

- ANALYZED
- ARGUMENT
- CODE
- EXPRESSION
- NOT
- RETURN
- RIGHT

Exit

A call to *Exit*



List of available properties :

- args_max
- args_min
- code
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- rank

List of possible tokens :

- T_COMMA
- T_EXIT
- T_OPEN_PARENTHESIS

List of outgoing links :

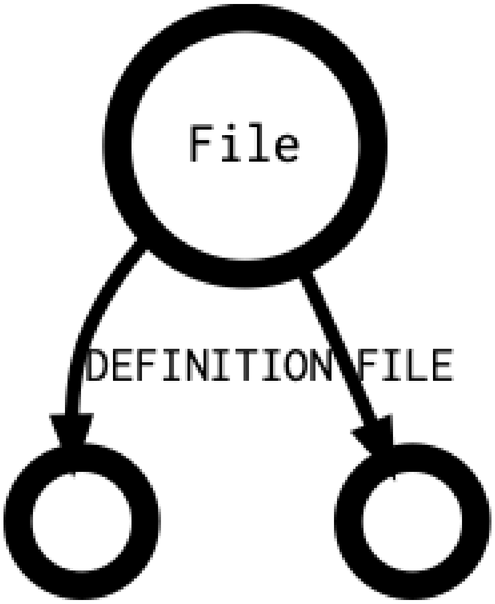
- ARGUMENT

List of incoming links :

- ANALYZED
- EXPRESSION
- RIGHT

File

A file, containing the PHP source code.



List of available properties :

- code
- ctype1_size
- fullcode
- lccode
- line

List of possible tokens :

- T_FILENAME

List of outgoing links :

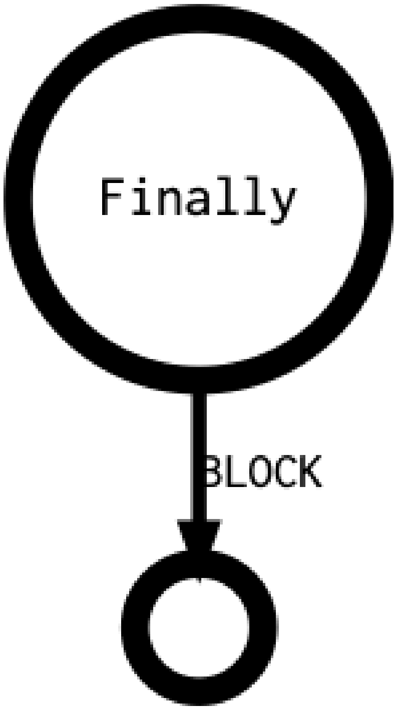
- DEFINITION
- FILE

List of incoming links :

- ANALYZED
- PROJECT

Finally

A finally clause in a try/catch command.



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line

List of possible tokens :

- T_TRY

List of outgoing links :

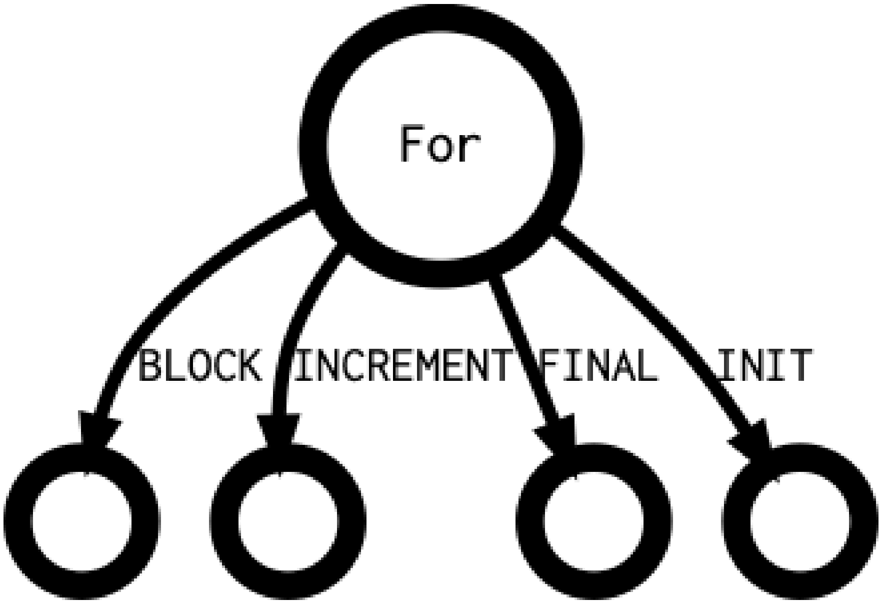
- BLOCK

List of incoming links :

- FINALLY

For

A for loop. For example : *for($i = 0; $i < 10; ++$i) { }*



List of available properties :

- alternative
- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_CLOSE_CURLY
- T_OPEN_TAG
- T_SEMICOLON
- 1

List of outgoing links :

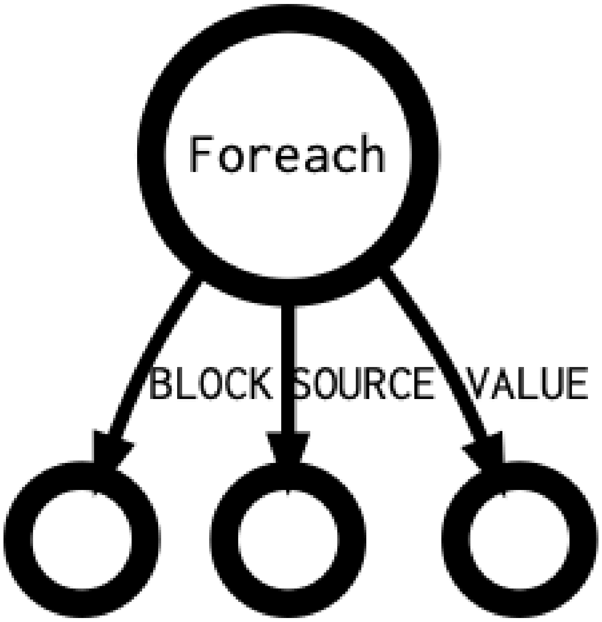
- BLOCK
- FINAL
- INCREMENT
- INIT

List of incoming links :

- ANALYZED
- EXPRESSION

Foreach

A foreach loop.



List of available properties :

- alternative
- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_FOREACH

List of outgoing links :

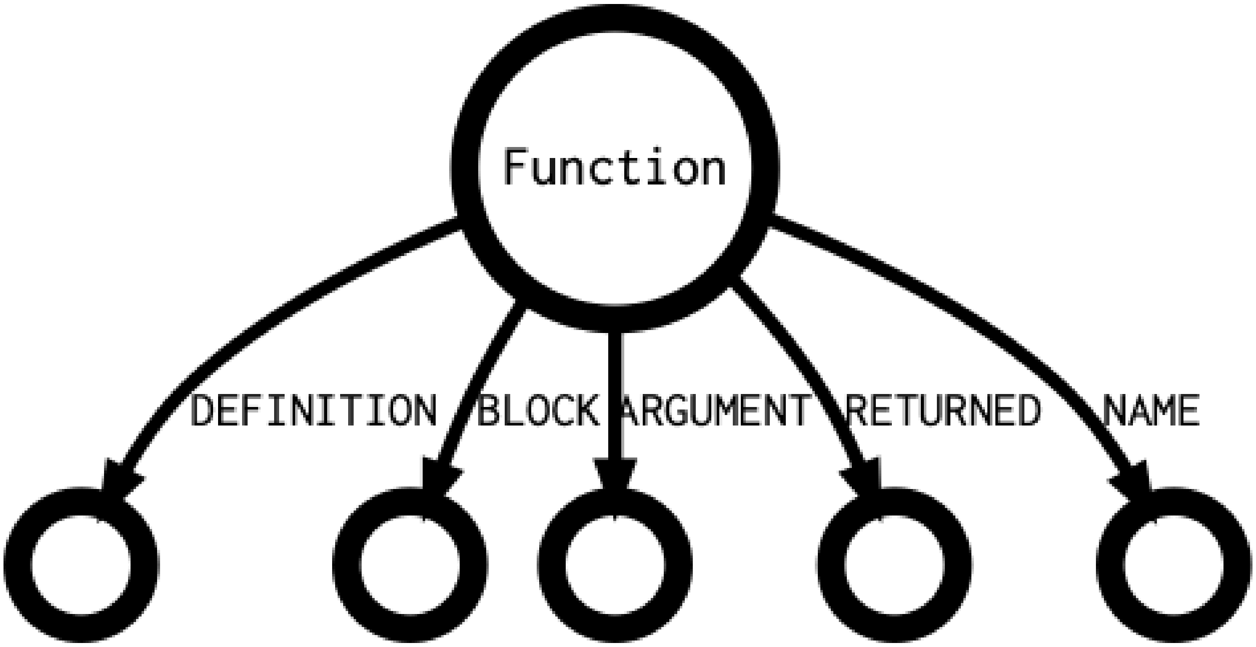
- BLOCK
- SOURCE
- VALUE

List of incoming links :

- ANALYZED
- EXPRESSION

Function

A function definition



List of available properties :

- args_max
- args_min
- code
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- nullable
- rank
- reference

List of possible tokens :

- T_FUNCTION

List of outgoing links :

- ARGUMENT
- BLOCK
- DEFINITION

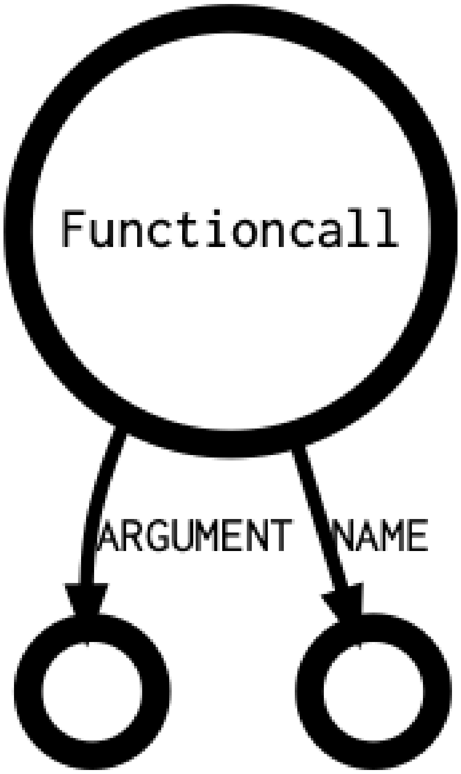
- NAME
- RETURNED
- RETURNTYPE

List of incoming links :

- ANALYZED
- EXPRESSION

Functioncall

A call to a function.



List of available properties :

- aliased
- args_max
- args_min
- code
- constant
- count
- ctype1
- ctype1_size
- enclosing
- fullcode

- fullnspath
- lccode
- line
- noDelimiter
- noscream
- rank
- reference
- variadic

List of possible tokens :

- T_CLOSE_BRACKET
- T_CLOSE_PARENTHESIS
- T_CONSTANT_ENCAPSED_STRING
- T_CURLY_OPEN
- T_DOLLAR
- T_DOUBLE_COLON
- T_NS_SEPARATOR
- T_OBJECT_OPERATOR
- T_OPEN_BRACKET
- T_OPEN_PARENTHESIS
- T_STRING
- T_VARIABLE

List of outgoing links :

- ARGUMENT
- NAME

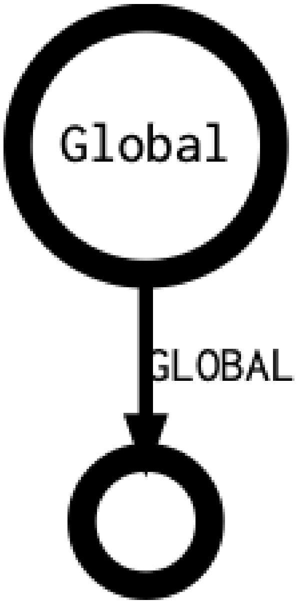
List of incoming links :

- ANALYZED
- APPEND
- ARGUMENT
- CASE
- CAST
- CODE
- CONCAT
- CONDITION
- DEFINITION
- ELSE
- EXPRESSION

- INDEX
- LEFT
- NAME
- NEW
- NOT
- OBJECT
- RETURN
- RIGHT
- SIGN
- SOURCE
- THEN
- THROW
- VALUE
- VARIABLE
- YIELD

Global

An expression with the global keyword. For example, *global $x, $y*.



List of available properties :

- code
- count
- ctype1
- ctype1_size

- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_GLOBAL

List of outgoing links :

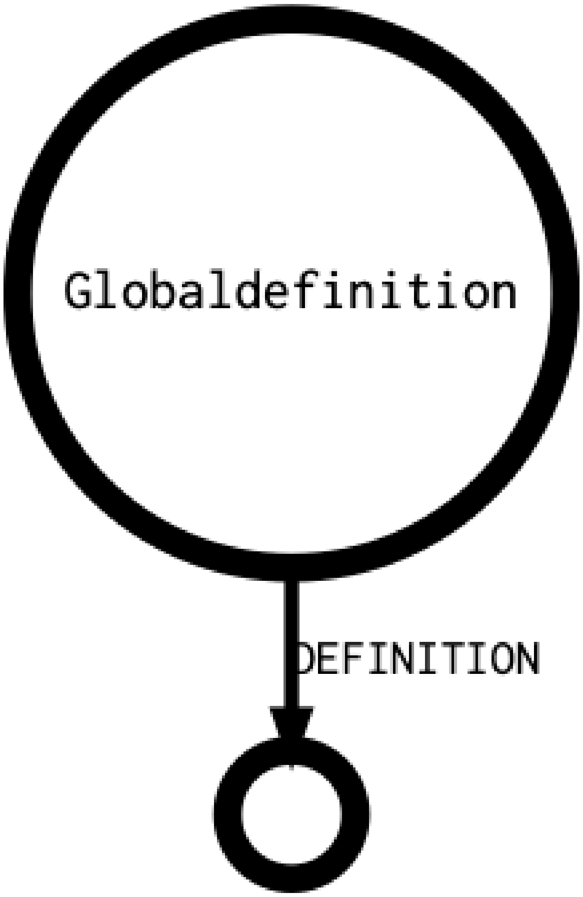
- GLOBAL

List of incoming links :

- ANALYZED
- EXPRESSION

Globaldefinition

A definition of a global variable, inside a global expression. For example, in *global $x = 1, $y*, *$x = 1* and *$y* are Globaldefinition.



List of available properties :

- code

- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_VARIABLE

List of outgoing links :

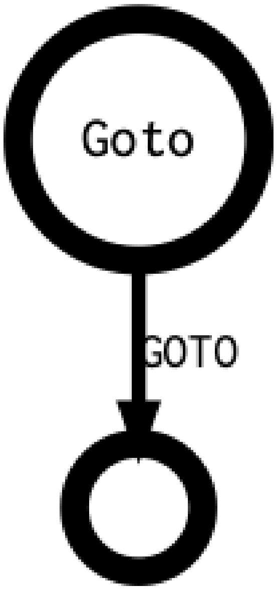
- DEFINITION

List of incoming links :

- ANALYZED
- DEFINITION
- GLOBAL

Goto

The goto expression.



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode

- line
- rank

List of possible tokens :

- T_GOTO

List of outgoing links :

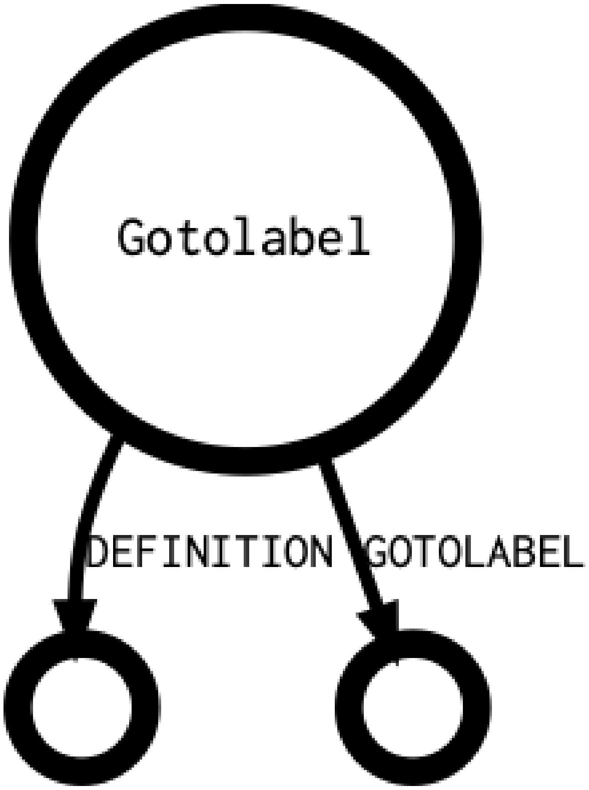
- GOTO

List of incoming links :

- DEFINITION
- EXPRESSION

Gotolabel

A target destination for a goto expression.



List of available properties :

- code
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_COLON

List of outgoing links :

- DEFINITION
- GOTOLABEL

List of incoming links :

- EXPRESSION

Halt

The *__halt_compiler* command.

List of available properties :

- code
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_HALT_COMPILER

List of outgoing links :

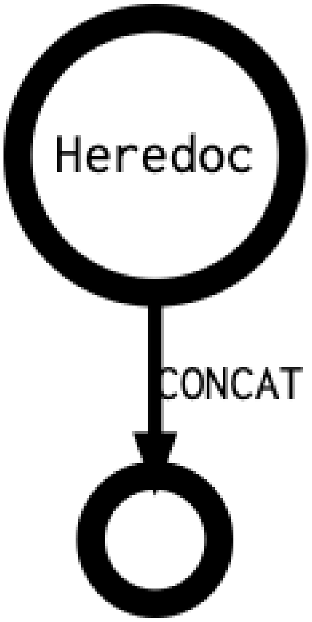
-

List of incoming links :

- EXPRESSION

Heredoc

A Heredoc or Nowdoc string



List of available properties :

- binaryString
- boolean
- code
- count
- ctype1
- ctype1_size
- delimiter
- fullcode
- heredoc
- intval
- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- T_START_HEREDOC

List of outgoing links :

- CONCAT

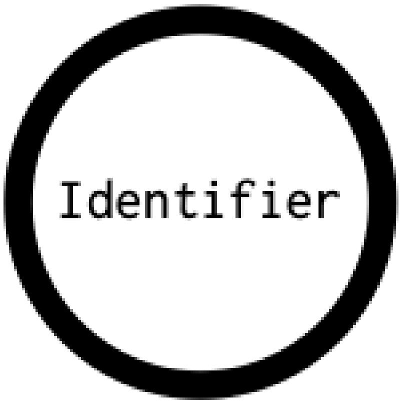
List of incoming links :

- ANALYZED
- ARGUMENT
- CONCAT

- DEFAULT
- RETURN
- RIGHT
- VALUE

Identifier

A name for a constant or a class. For example : *$x instanceof Y*, ‘echo PHP_INT_MAX‘, *new Y*



List of available properties :

- aliased
- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- fullnspath
- intval
- isNull
- lccode
- line
- noDelimiter
- noscream
- rank
- reference

List of possible tokens :

- T_ARRAY
- T_CALLABLE

- T_CONST
- T_FUNCTION
- T_STRING

List of outgoing links :

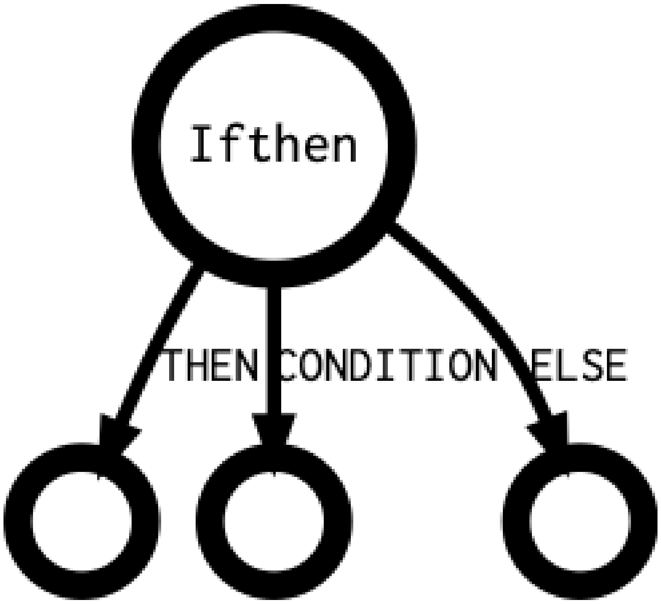
- DEFINITION

List of incoming links :

- ANALYZED
- ARGUMENT
- AS
- CASE
- CAST
- CLASS
- CODE
- CONCAT
- CONDITION
- CONST
- DEFAULT
- DEFINITION
- ELSE
- EXPRESSION
- FUNCTION
- INDEX
- INSTEADOF
- LEFT
- MEMBER
- NAME
- NEW
- NOT
- RETURN
- RIGHT
- SIGN
- THEN
- TYPEHINT
- VALUE
- VARIABLE

Ifthen

A if/then/else structure.



List of available properties :

- alternative
- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_ELSEIF
- T_IF

List of outgoing links :

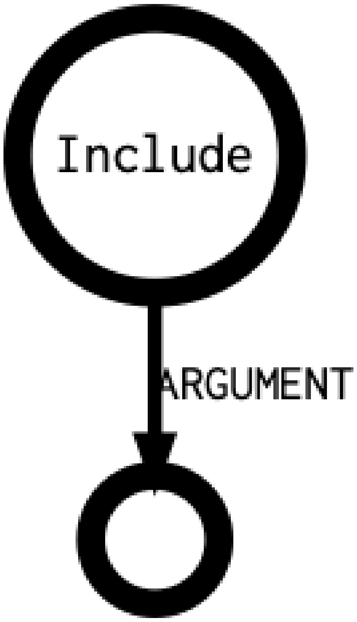
- CONDITION
- ELSE
- THEN

List of incoming links :

- ANALYZED
- ELSE
- EXPRESSION

Include

A inclusion, with *require* or *include*, with *_once* or not.



List of available properties :

- code
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- noscream
- rank

List of possible tokens :

- T_INCLUDE
- T_INCLUDE_ONCE
- T_REQUIRE
- T_REQUIRE_ONCE

List of outgoing links :

- ARGUMENT

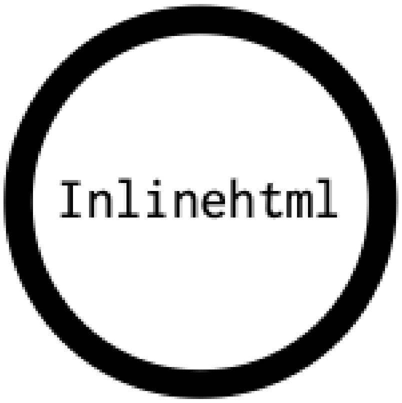
List of incoming links :

- ANALYZED

- ARGUMENT
- CODE
- CONDITION
- EXPRESSION
- NOT
- RETURN
- RIGHT
- THEN

Inlinehtml

Raw text, in the middle of a PHP script. For example : ‘‘++$a; ?>RAW TEXT<?php ++$b; ‘‘



List of available properties :

- code
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_INLINE_HTML
- 1

List of outgoing links :

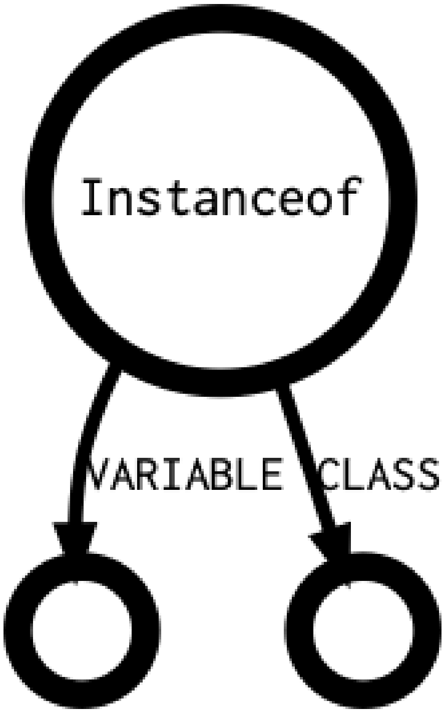
-

List of incoming links :

- ANALYZED
- EXPRESSION

Instanceof

A *instanceof* expression



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_INSTANCEOF

List of outgoing links :

- CLASS
- VARIABLE

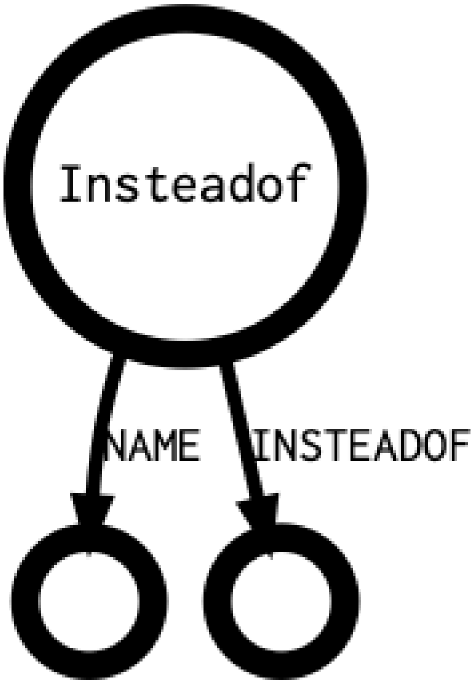
List of incoming links :

- ANALYZED
- ARGUMENT
- CASE
- CODE

- CONDITION
- INDEX
- LEFT
- NOT
- RETURN
- RIGHT

Insteadof

A *insteadof* expression



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_INSTEADOF

List of outgoing links :

- INSTEADOF

- NAME

List of incoming links :

- EXPRESSION

Integer

An Integer literal, positive or negative.

List of available properties :

- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- fullnspath
- intval
- lccode
- line
- noDelimiter
- rank
- variadic

List of possible tokens :

- T_LNUMBER
- T_NUM_STRING

List of outgoing links :

-

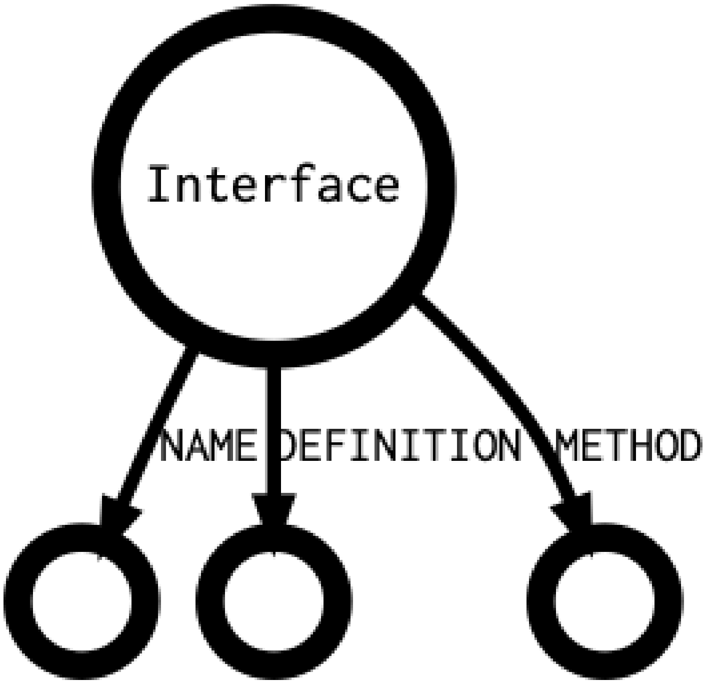
List of incoming links :

- ANALYZED
- ARGUMENT
- BREAK

- CASE
- CAST
- CLONE
- CODE
- CONCAT
- CONDITION
- CONTINUE
- DEFAULT
- ELSE
- INDEX
- LEFT
- NAME
- NOT
- RETURN
- RIGHT
- THEN
- THROW
- VALUE
- YIELD

Interface

An interface definition



List of available properties :

- code
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- rank

List of possible tokens :

- T_INTERFACE

List of outgoing links :

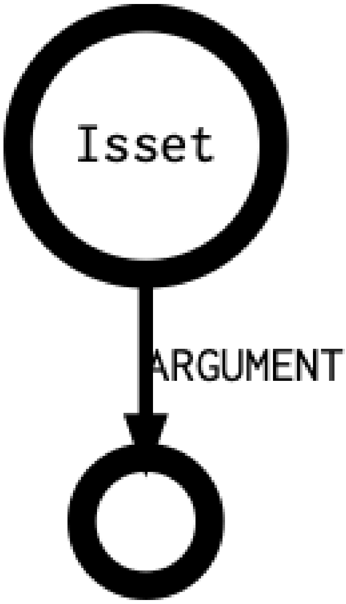
- CONST
- DEFINITION
- EXTENDS
- MAGICMETHOD
- METHOD
- NAME

List of incoming links :

- ANALYZED
- EXPRESSION

Isset

A call to *isset*



List of available properties :

- args_max
- args_min
- code
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- rank

List of possible tokens :

- T_ISSET

List of outgoing links :

- ARGUMENT

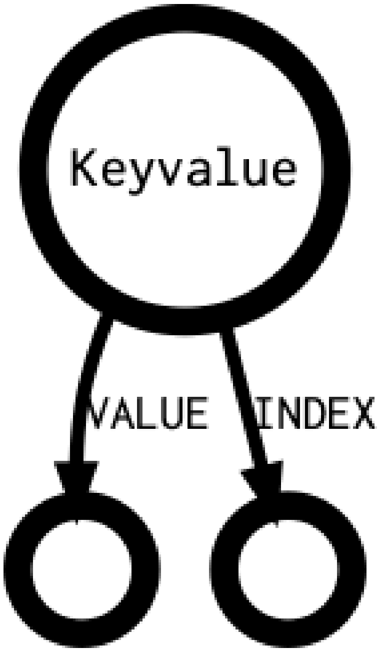
List of incoming links :

- ANALYZED
- ARGUMENT
- CASE
- CODE

- CONDITION
- EXPRESSION
- LEFT
- NOT
- RETURN
- RIGHT

Keyvalue

An expression with the => operator : for arrays or foreach() instructions.



List of available properties :

- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_DOUBLE_ARROW

List of outgoing links :

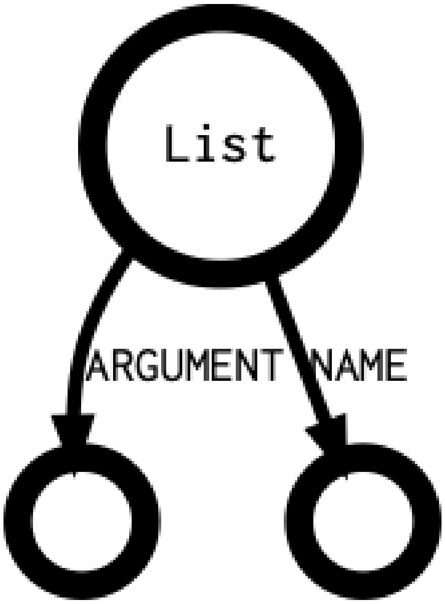
- INDEX
- VALUE

List of incoming links :

- ARGUMENT
- VALUE
- YIELD

List

The list() or [] call when on the right of an assignation.



List of available properties :

- args_max
- args_min
- code
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- noscream
- rank

List of possible tokens :

- T_LIST
- T_OPEN_BRACKET

List of outgoing links :

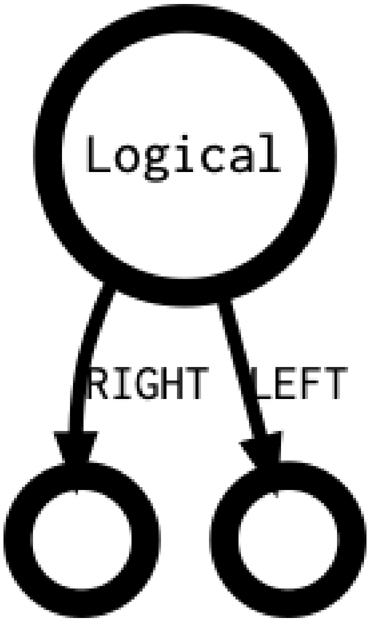
- ARGUMENT
- NAME

List of incoming links :

- ANALYZED
- ARGUMENT
- LEFT
- VALUE

Logical

A logical expression. This covers also bitwise operations. For example : *$a | $b*, *$a && $b*, *$a xor $b*.



List of available properties :

- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- intval
- lccode
- line

- noDelimiter
- rank

List of possible tokens :

- T_AND
- T_BOOLEAN_AND
- T_BOOLEAN_OR
- T_LOGICAL_AND
- T_LOGICAL_OR
- T_LOGICAL_XOR
- T_OR
- T_XOR

List of outgoing links :

- LEFT
- RIGHT

List of incoming links :

- ANALYZED
- ARGUMENT
- CASE
- CODE
- CONDITION
- DEFAULT
- ELSE
- EXPRESSION
- INDEX
- LEFT
- RETURN
- RIGHT
- THEN
- VALUE

Magicconstant

A PHP magic constant. For example : __FILE__ or __class__.

List of available properties :

- boolean
- code
- ctype1
- ctype1_size
- fullcode
- intval
- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- T_CLASS_C
- T_DIR
- T_FILE
- T_FUNC_C
- T_LINE
- T_METHOD_C
- T_NS_C
- T_TRAIT_C
- 1

List of outgoing links :

-

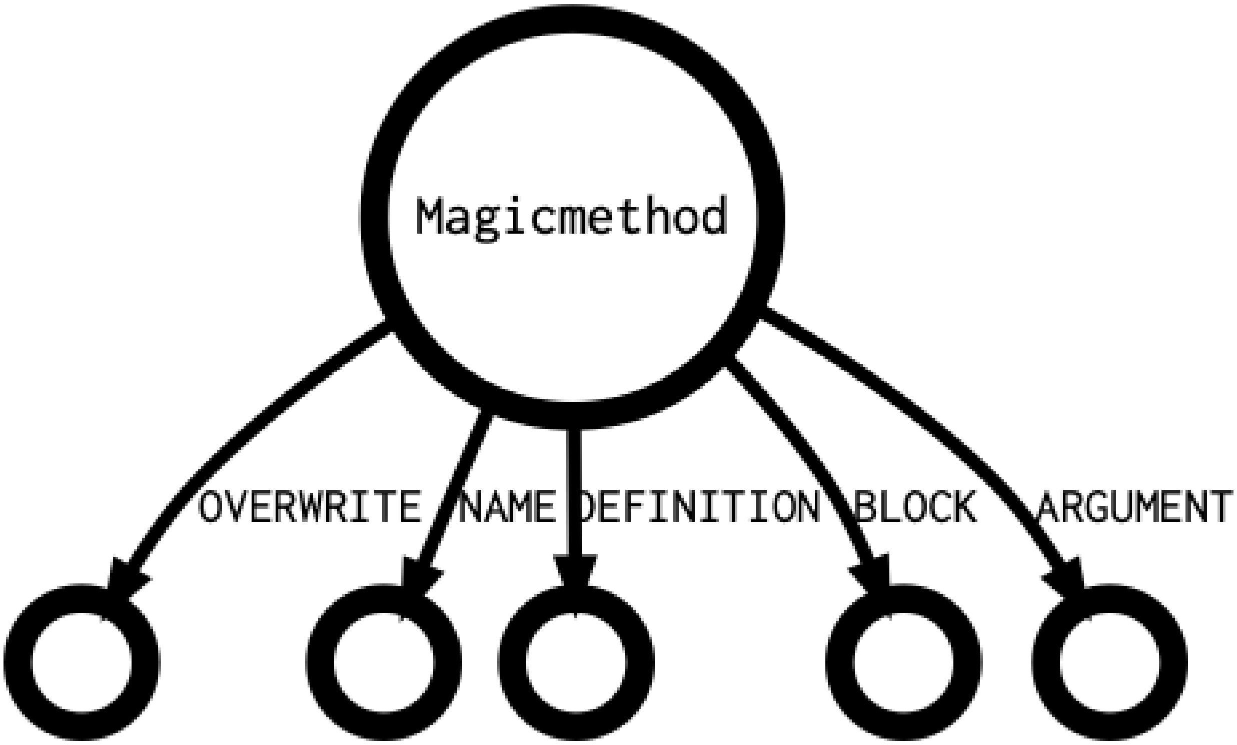
List of incoming links :

- ANALYZED

- ARGUMENT
- CODE
- CONCAT
- DEFAULT
- ELSE
- INDEX
- LEFT
- RETURN
- RIGHT
- THEN
- VALUE

Magicmethod

A special PHP method in a class. For example, *__clone()*, *__construct()*, *__get()*, …



List of available properties :

- abstract
- args_max
- args_min
- code
- count

- ctype1
- ctype1_size
- final
- fullcode
- fullnspath
- lccode
- line
- rank
- static
- visibility

List of possible tokens :

- T_FUNCTION
- 1

List of outgoing links :

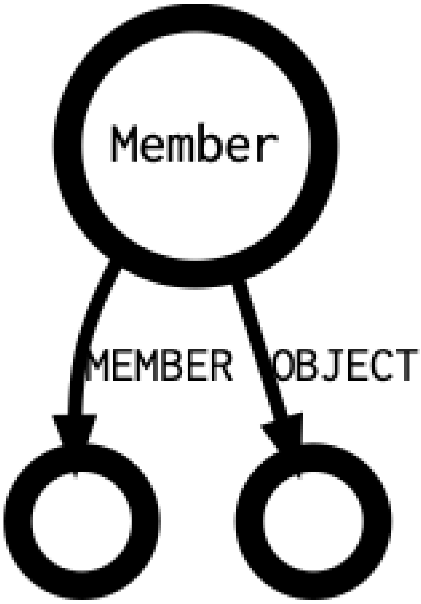
- ARGUMENT
- BLOCK
- DEFINITION
- NAME
- OVERWRITE
- RETURNED
- RETURNTYPE

List of incoming links :

- ANALYZED
- MAGICMETHOD
- OVERWRITE

Member

A reference to a member of an object. For example, *$object->member*.



List of available properties :

- code
- ctype1
- ctype1_size
- enclosing
- fullcode
- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- T_CURLY_OPEN
- T_OBJECT_OPERATOR
- T_QUOTE
- T_START_HEREDOC

List of outgoing links :

- MEMBER
- OBJECT

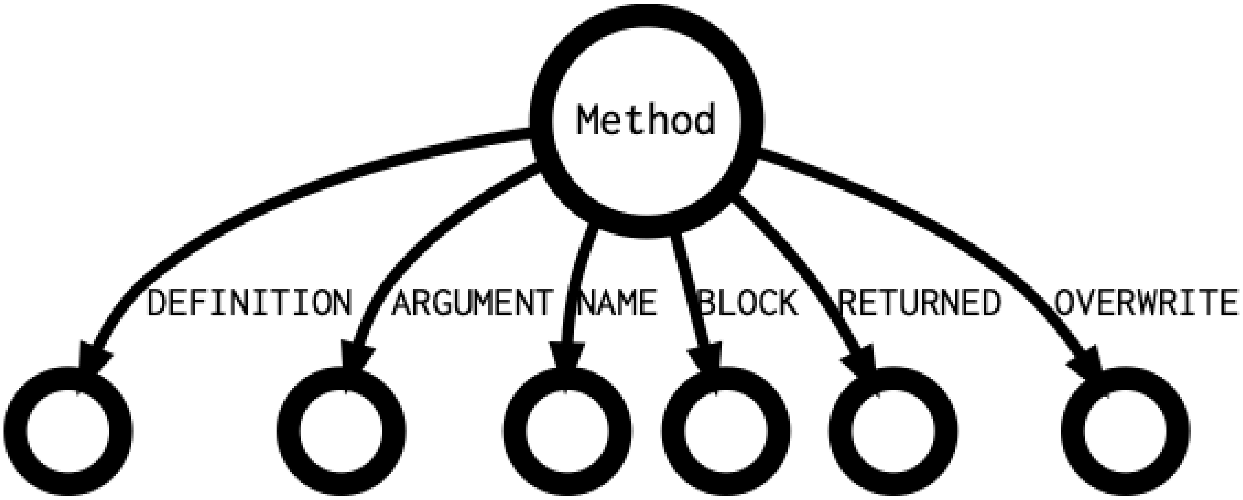
List of incoming links :

- ANALYZED
- APPEND
- ARGUMENT
- CAST

- CLASS
- CLONE
- CODE
- CONCAT
- CONDITION
- DEFINITION
- ELSE
- EXPRESSION
- INDEX
- LEFT
- NAME
- NEW
- NOT
- OBJECT
- POSTPLUSPLUS
- PREPLUSPLUS
- RETURN
- RIGHT
- SIGN
- SOURCE
- THEN
- THROW
- VALUE
- VARIABLE
- YIELD

Method

A method definition in a class.



List of available properties :

- abstract
- args_max
- args_min
- code
- count
- ctype1
- ctype1_size
- final
- fullcode
- fullnspath
- lccode
- line
- nullable
- rank
- reference
- static
- visibility

List of possible tokens :

- T_FUNCTION
- 1

List of outgoing links :

- ARGUMENT
- BLOCK
- DEFINITION

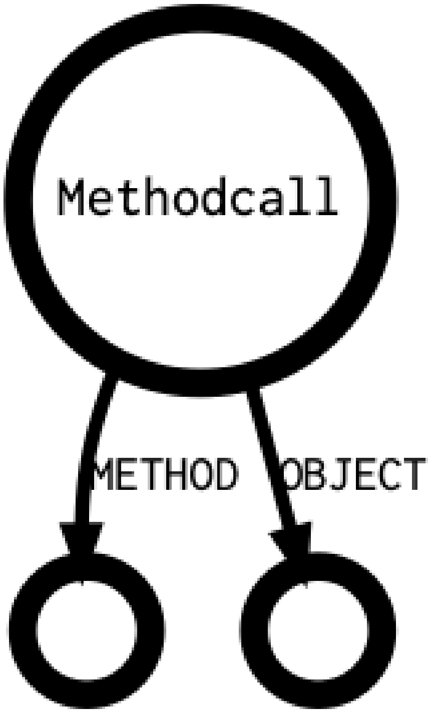
- NAME
- OVERWRITE
- RETURNED
- RETURNTYPE

List of incoming links :

- ANALYZED
- METHOD
- OVERWRITE

Methodcall

A non-static call to a method. For example, *$a->method();*



List of available properties :

- code
- ctype1
- ctype1_size
- enclosing
- fullcode
- lccode
- line
- noDelimiter
- rank

- variadic

List of possible tokens :

- T_CURLY_OPEN
- T_OBJECT_OPERATOR

List of outgoing links :

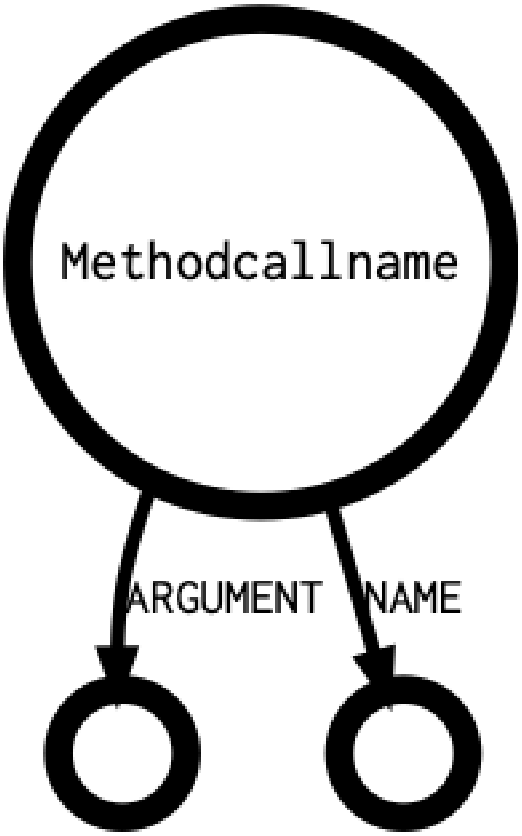
- METHOD
- OBJECT

List of incoming links :

- ANALYZED
- APPEND
- ARGUMENT
- CASE
- CAST
- CLONE
- CODE
- CONCAT
- CONDITION
- DEFINITION
- ELSE
- EXPRESSION
- INDEX
- LEFT
- NAME
- NEW
- NOT
- OBJECT
- RETURN
- RIGHT
- SIGN
- SOURCE
- THEN
- THROW
- VALUE
- VARIABLE
- YIELD

Methodcallname

The name of the method in a methodcall



List of available properties :

- args_max
- args_min
- code
- count
- ctype1
- ctype1_size
- fullcode
- lccode
- line

List of possible tokens :

- T_CLOSE_BRACKET
- T_CONSTANT_ENCAPSED_STRING
- T_DOLLAR
- T_NEW
- T_STRING

- T_VARIABLE

List of outgoing links :

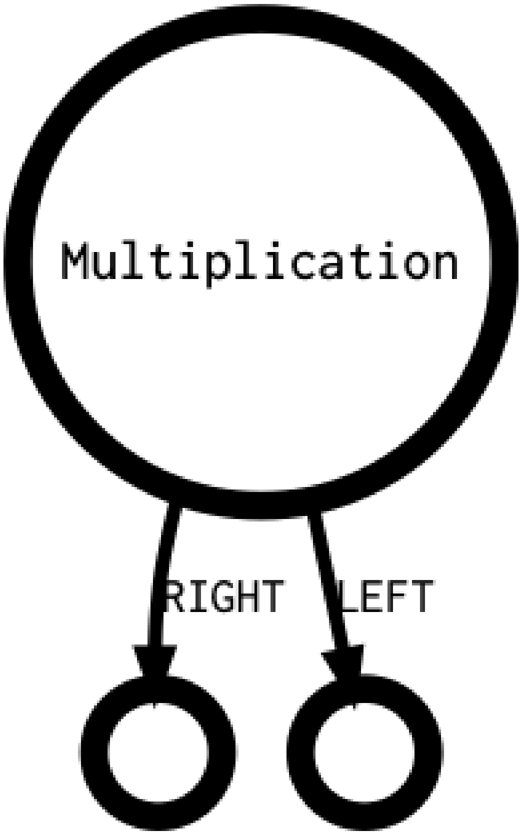
- ARGUMENT
- NAME

List of incoming links :

- ANALYZED
- METHOD

Multiplication

A multiplication *, division / or modulo % operation.



List of available properties :

- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- intval

- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- T_PERCENTAGE
- T_SLASH
- T_STAR

List of outgoing links :

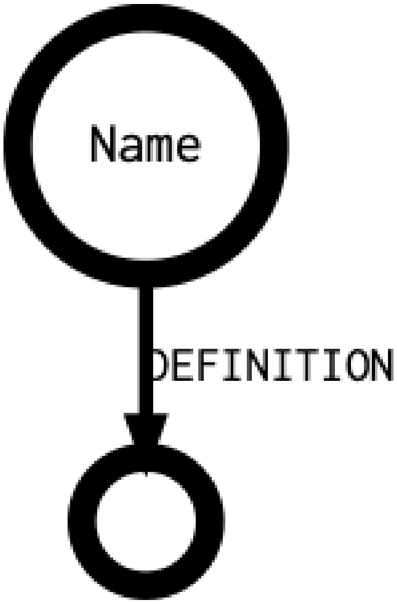
- LEFT
- RIGHT

List of incoming links :

- ANALYZED
- ARGUMENT
- CODE
- CONCAT
- CONDITION
- DEFAULT
- ELSE
- EXPRESSION
- INDEX
- LEFT
- RETURN
- RIGHT
- SIGN
- THEN
- VALUE

Name

The name of a structure : name of a class, method, interface, trait, interface.



List of available properties :

- aliased
- code
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line

List of possible tokens :

- T_ABSTRACT
- T_CLASS
- T_INSTANCEOF
- T_LIST
- T_NEW
- T_OPEN_TAG_WITH_ECHO
- T_PRINT
- T_PRIVATE
- T_PUBLIC
- T_STRING
- T_THROW

List of outgoing links :

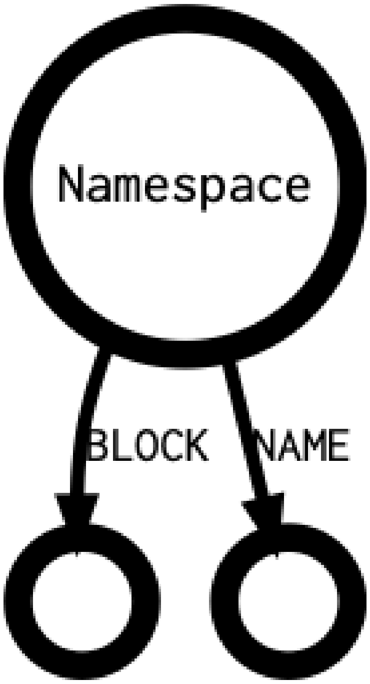
- DEFINITION

List of incoming links :

- ANALYZED
- CONSTANT
- DEFINITION
- GOTO
- GOTOLABEL
- MEMBER
- METHOD
- NAME

Namespace

A namespace declaration



List of available properties :

- code
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- rank

List of possible tokens :

- T_NAMESPACE

List of outgoing links :

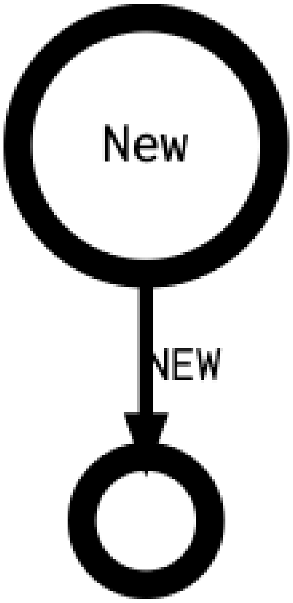
- BLOCK
- NAME

List of incoming links :

- ANALYZED
- EXPRESSION

New

An instantiation expression, with *new ClassName()*.



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- noscream
- rank
- variadic

List of possible tokens :

- T_NEW

List of outgoing links :

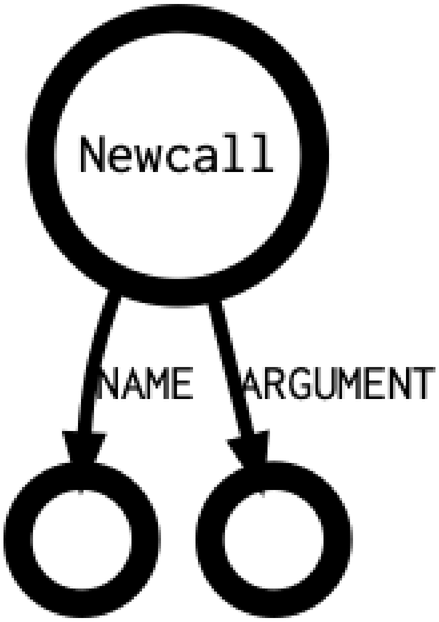
- NEW

List of incoming links :

- ANALYZED
- ARGUMENT
- CAST
- CLONE
- CODE
- CONCAT
- CONDITION
- DEFINITION
- ELSE
- EXPRESSION
- INDEX
- LEFT
- RETURN
- RIGHT
- SOURCE
- THEN
- THROW
- VALUE
- VARIABLE
- YIELD

Newcall

The functioncall in a New expression. For example, in ‘‘new foo()‘, *foo()* is the Newcall.



List of available properties :

- absolute
- aliased
- args_max
- args_min
- code
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- noscream
- rank

List of possible tokens :

- T_DOLLAR
- T_LIST
- T_NS_SEPARATOR
- T_STATIC
- T_STRING
- T_VARIABLE

List of outgoing links :

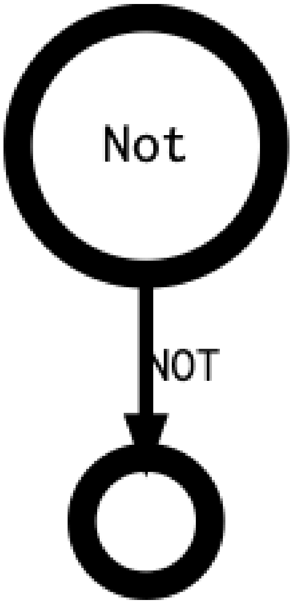
- ARGUMENT
- DEFINITION
- NAME

List of incoming links :

- ANALYZED
- ARGUMENT
- CAST
- CLASS
- CODE
- CONCAT
- CONDITION
- DEFINITION
- ELSE
- EXPRESSION
- INDEX
- LEFT
- NAME
- NEW
- NOT
- OBJECT
- RETURN
- RIGHT
- SIGN
- THEN
- TYPEHINT
- VALUE
- VARIABLE

Not

A call to *!* or ~.



List of available properties :

- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- intval
- lccode
- line
- noDelimiter
- noscream
- rank

List of possible tokens :

- T_BANG
- T_TILDE

List of outgoing links :

- NOT

List of incoming links :

- ANALYZED
- ARGUMENT
- CASE
- CAST

- CODE
- CONDITION
- ELSE
- EXPRESSION
- LEFT
- NOT
- RETURN
- RIGHT
- THEN
- VALUE

Nsname

A fully qualified name, including ‘*. For example, ‘strtolower*, *ABC*, … w

List of available properties :

- absolute
- alias
- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- fullnspath
- intval
- isNull
- lccode
- line
- noDelimiter
- origin

- rank
- reference

List of possible tokens :

- T_ARRAY
- T_CALLABLE
- T_NS_SEPARATOR
- T_STRING
- 1

List of outgoing links :

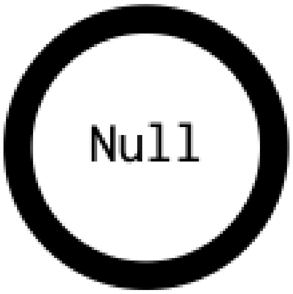
- DEFINITION

List of incoming links :

- ANALYZED
- ARGUMENT
- CLASS
- CONCAT
- DEFAULT
- DEFINITION
- EXTENDS
- IMPLEMENTS
- INDEX
- LEFT
- NAME
- NEW
- RETURNTYPE
- RIGHT
- TYPEHINT
- USE
- VALUE

Null

The *Null* value



List of available properties :

- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- fullnspath
- intval
- isNull
- lccode
- line
- noDelimiter
- rank
- variadic

List of possible tokens :

- T_STRING

List of outgoing links :

-

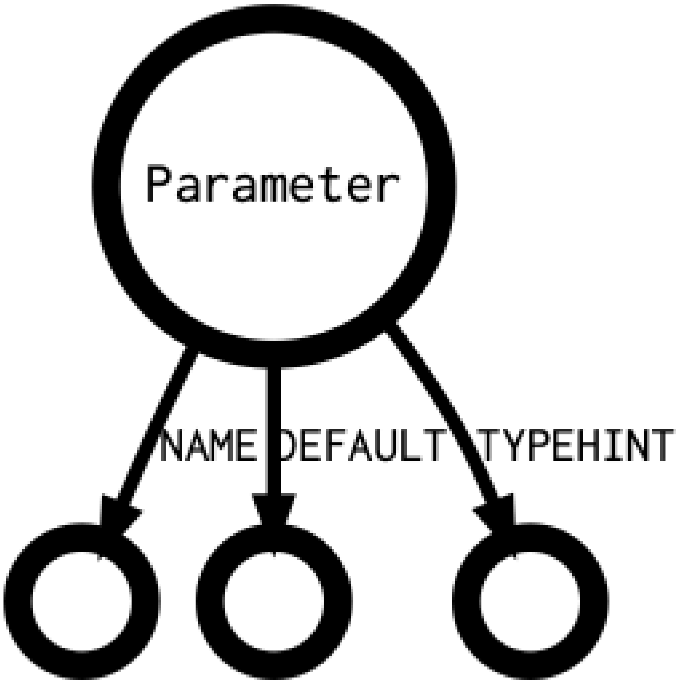
List of incoming links :

- ANALYZED
- ARGUMENT
- CASE
- CAST
- CODE
- CONDITION
- DEFAULT
- ELSE
- INDEX
- LEFT

- RETURN
- RIGHT
- THEN
- VALUE

Parameter

A parameter definition, in a function or method definition. When called, it becomes an argument.



List of available properties :

- code
- ctype1_size
- fullcode
- lccode
- line
- nullable
- rank
- reference
- variadic

List of possible tokens :

- T_VARIABLE

List of outgoing links :

- DEFAULT

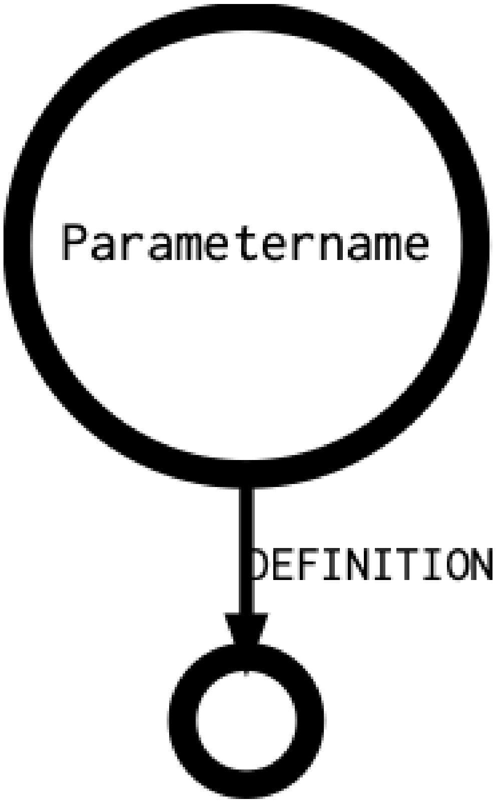
- DEFINITION
- NAME
- TYPEHINT

List of incoming links :

- ANALYZED
- ARGUMENT
- DEFINITION
- USE

Parametername

A Parametername



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_VARIABLE

List of outgoing links :

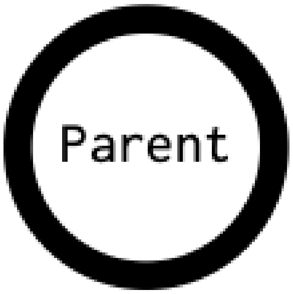
- DEFINITION

List of incoming links :

- ANALYZED
- GLOBAL
- NAME

Parent

The parent keyword, when it is used to refer to the parent class.



List of available properties :

- boolean
- code
- ctype1
- ctype1_size
- fullcode
- fullnspath
- intval
- lccode
- line
- noscream

List of possible tokens :

- T_STRING
- 1

List of outgoing links :

-

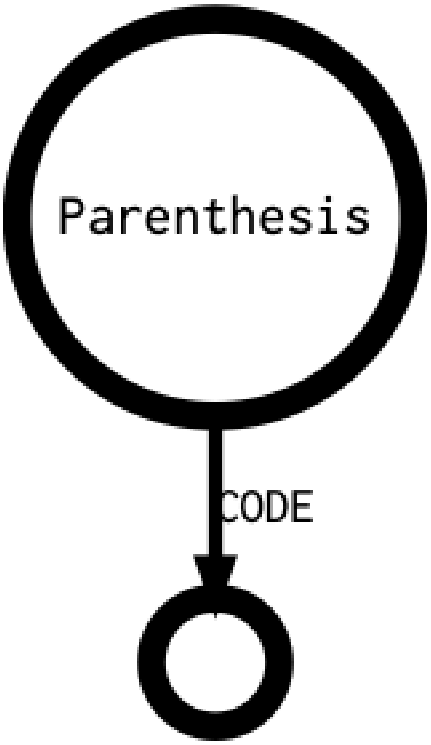
List of incoming links :

- ANALYZED

- CLASS
- DEFINITION
- NEW
- RETURNTYPE
- TYPEHINT

Parenthesis

A Parenthesis expression. This is not a syntactic parenthesis, like in a switch or functioncall.



List of available properties :

- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- intval
- isNull
- lccode
- line
- noDelimiter

- noscream
- rank

List of possible tokens :

- T_OPEN_PARENTHESIS

List of outgoing links :

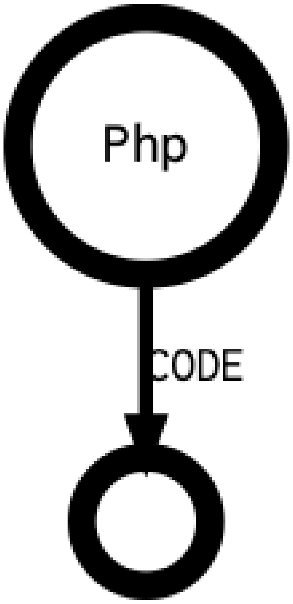
- CODE

List of incoming links :

- ANALYZED
- ARGUMENT
- CASE
- CAST
- CLONE
- CODE
- CONCAT
- CONDITION
- ELSE
- EXPRESSION
- INDEX
- LEFT
- NAME
- NOT
- OBJECT
- RETURN
- RIGHT
- SIGN
- SOURCE
- THEN
- THROW
- VALUE
- VARIABLE

Php

A PHP script, inside its tags. This exclude the following and previous raw text in a PHP file.



List of available properties :

- close_tag
- code
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_OPEN_TAG

List of outgoing links :

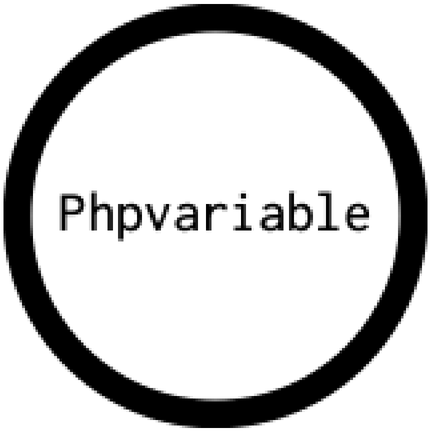
- CODE

List of incoming links :

- EXPRESSION

Phpvariable

A PHP reserved variable, such as *$_GET*, *$_POST*, *$GLOBALS*, etc.



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- noDelimiter
- rank
- reference
- variadic

List of possible tokens :

- T_VARIABLE

List of outgoing links :

-

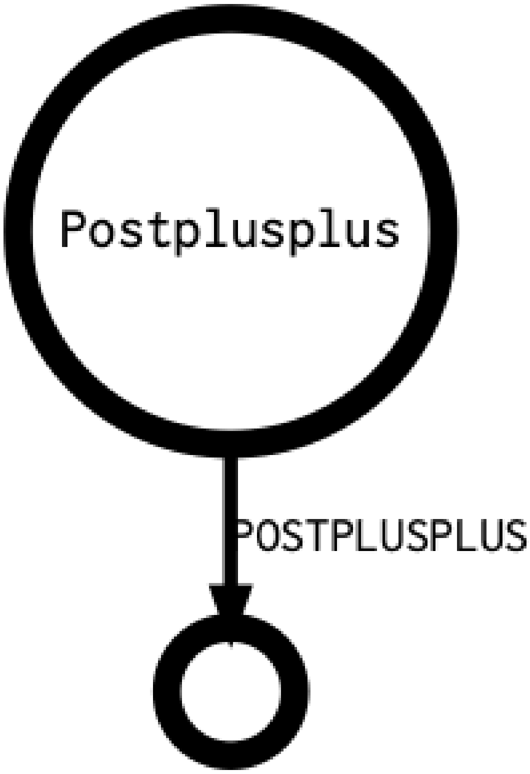
List of incoming links :

- ANALYZED
- APPEND
- ARGUMENT
- CONCAT
- CONDITION
- DEFINITION
- ELSE
- LEFT
- RIGHT
- SOURCE
- THEN

- VALUE
- VARIABLE

Postplusplus

$i++‘ expression



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_DEC
- T_INC

List of outgoing links :

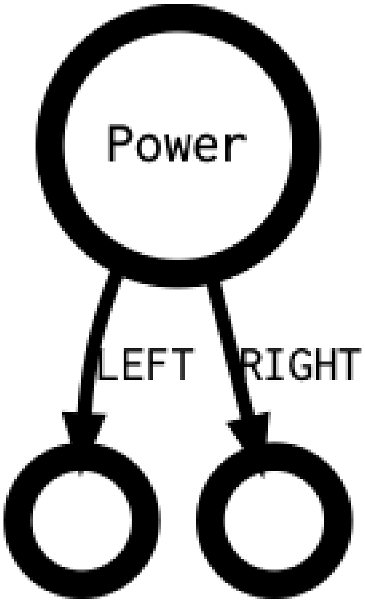
- POSTPLUSPLUS

List of incoming links :

- ANALYZED
- ARGUMENT
- CODE
- CONCAT
- CONDITION
- ELSE
- EXPRESSION
- INDEX
- LEFT
- NOT
- RETURN
- RIGHT
- THEN

Power

The power operator, **.



List of available properties :

- boolean
- code
- ctype1
- ctype1_size
- fullcode
- intval

- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- T_POW

List of outgoing links :

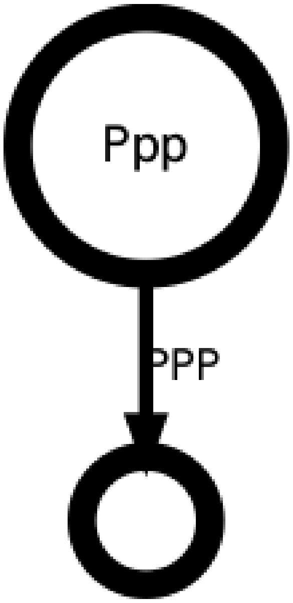
- LEFT
- RIGHT

List of incoming links :

- ARGUMENT
- CODE
- VALUE

Ppp

A properties declaration, in a class or a trait. For example : *private $x, $y = 2;*



List of available properties :

- code
- count
- ctype1
- ctype1_size
- fullcode
- lccode

- line
- rank
- static
- visibility

List of possible tokens :

- T_PRIVATE
- T_PROTECTED
- T_PUBLIC
- T_STATIC
- T_VAR
- 1

List of outgoing links :

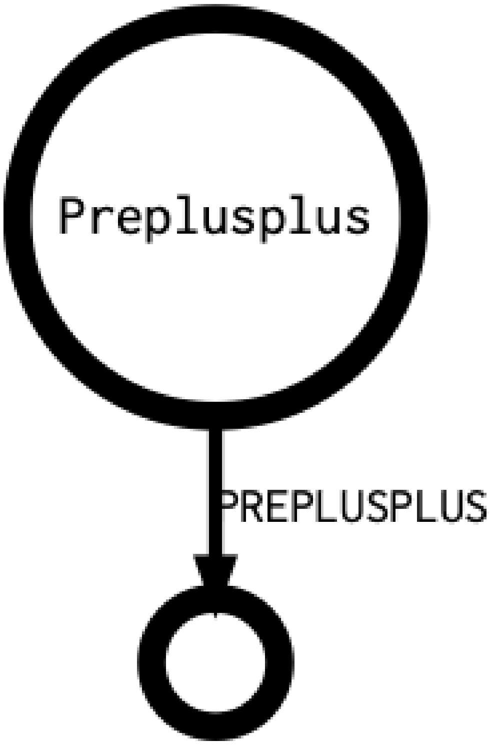
- PPP

List of incoming links :

- ANALYZED
- PPP

Preplusplus

++ or – when it is before the variable.



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_DEC
- T_INC

List of outgoing links :

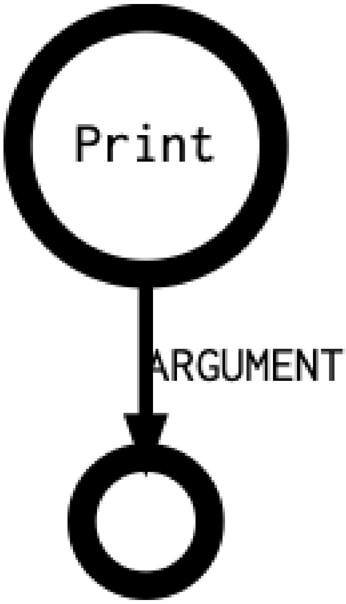
- PREPLUSPLUS

List of incoming links :

- ANALYZED
- ARGUMENT
- CODE
- CONCAT
- CONDITION
- ELSE
- EXPRESSION
- INDEX
- LEFT
- NOT
- RETURN
- RIGHT
- VALUE
- YIELD

Print

A call to the function print.



List of available properties :

- code
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- rank

List of possible tokens :

- T_PRINT

List of outgoing links :

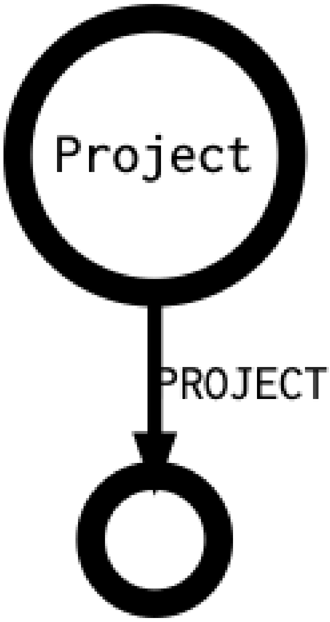
- ARGUMENT

List of incoming links :

- ANALYZED
- ELSE
- EXPRESSION
- RIGHT
- THEN

Project

The project node : the root above all File.



List of available properties :

- code
- ctype1_size
- fullcode
- lccode
- line

List of possible tokens :

- T_WHOLE

List of outgoing links :

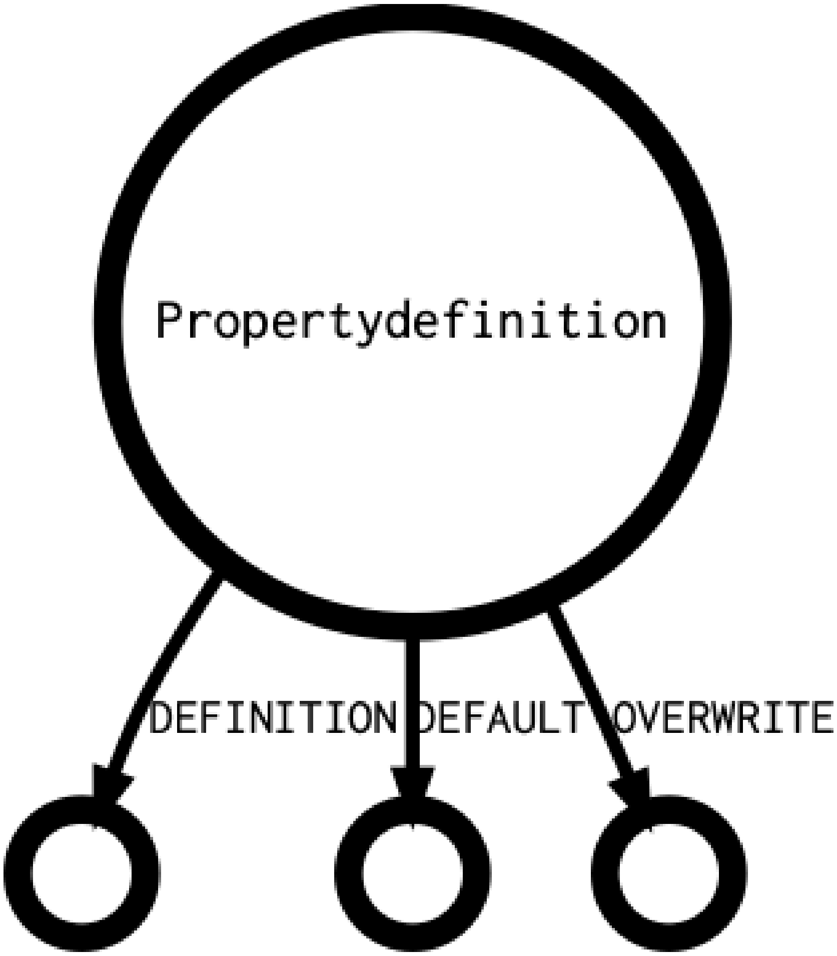
- PROJECT

List of incoming links :

-

Propertydefinition

A property definition. For example : `class x { private $property = 1; var $x; }`



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- propertyname
- rank

List of possible tokens :

- T_VARIABLE
- 1

List of outgoing links :

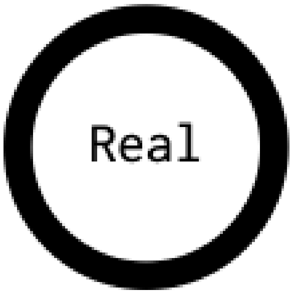
- DEFAULT
- DEFINITION
- OVERWRITE

List of incoming links :

- ANALYZED
- OVERWRITE
- PPP

Real

A float number



List of available properties :

- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- intval
- lccode
- line
- noDelimiter
- rank
- variadic

List of possible tokens :

- T_DNUMBER

List of outgoing links :

-

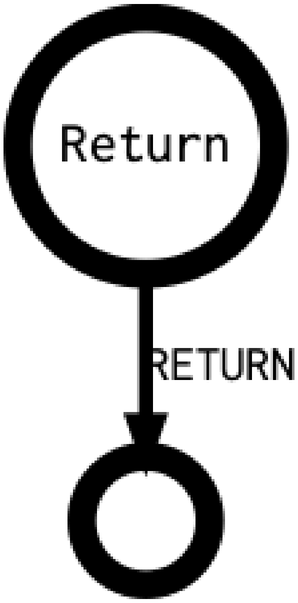
List of incoming links :

- ARGUMENT
- CAST
- CODE
- CONCAT
- DEFAULT

- ELSE
- INDEX
- LEFT
- RETURN
- RIGHT
- THEN
- VALUE

Return

The return expression.



List of available properties :

- code
- constant
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_RETURN

List of outgoing links :

- RETURN

List of incoming links :

- ANALYZED
- EXPRESSION

Self

The *self* keyword, as used inside a class.

List of available properties :

- boolean
- code
- ctype1
- ctype1_size
- fullcode
- fullnspath
- intval
- lccode
- line
- noscream
- reference

List of possible tokens :

- T_STRING
- 1

List of outgoing links :

-

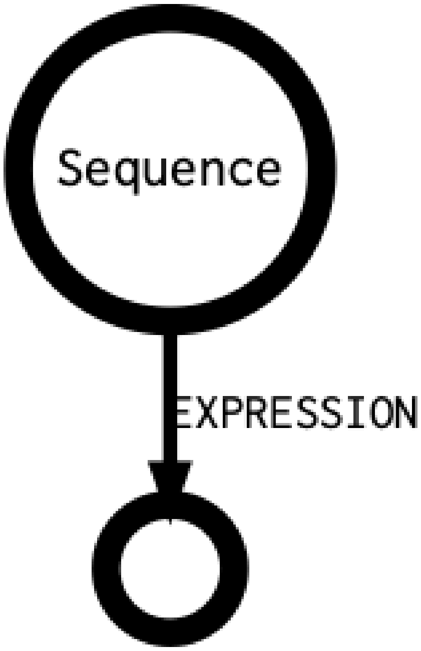
List of incoming links :

- ANALYZED
- CLASS
- DEFINITION
- NAME
- NEW

- RETURNTYPE
- TYPEHINT

Sequence

A virtual atom, that represents the sequence of expression, in a block.



List of available properties :

- boolean
- bracket
- code
- constant
- count
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank
- root

List of possible tokens :

- T_CLOSE_CURLY
- T_CLOSE_PARENTHESIS
- T_COLON

- T_CONSTANT_ENCAPSED_STRING
- T_INLINE_HTML
- T_OPEN_CURLY
- T_OPEN_TAG
- T_SEMICOLON
- T_SWITCH

List of outgoing links :

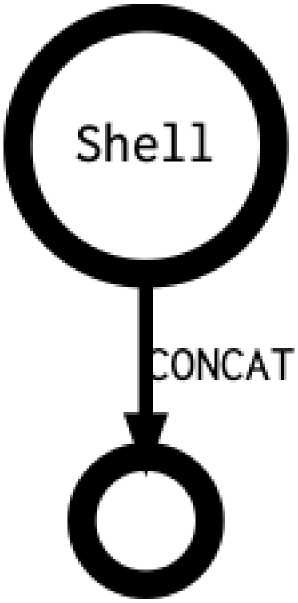
- EXPRESSION

List of incoming links :

- ANALYZED
- BLOCK
- CASES
- CODE
- ELSE
- EXPRESSION
- FILE
- FINAL
- INCREMENT
- INIT
- THEN

Shell

A shell, made with ticks ‘



List of available properties :

- code
- count
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- T_SHELL_QUOTE

List of outgoing links :

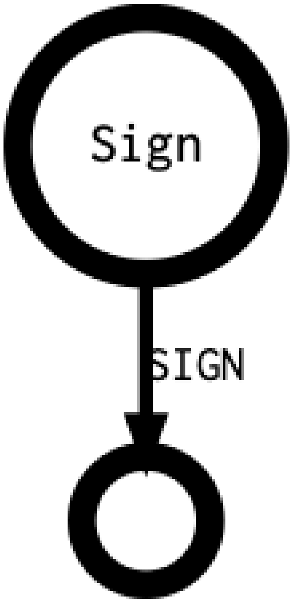
- CONCAT

List of incoming links :

- ARGUMENT
- EXPRESSION
- RIGHT

Sign

A Sign structure : when a -‘*or* ‘+ has been added before another expression. For example - *($a + $b)*.



List of available properties :

- code

- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_CLOSE_BRACKET
- T_CLOSE_PARENTHESIS
- T_CONSTANT_ENCAPSED_STRING
- T_LNUMBER
- T_STRING
- T_VARIABLE
- 1

List of outgoing links :

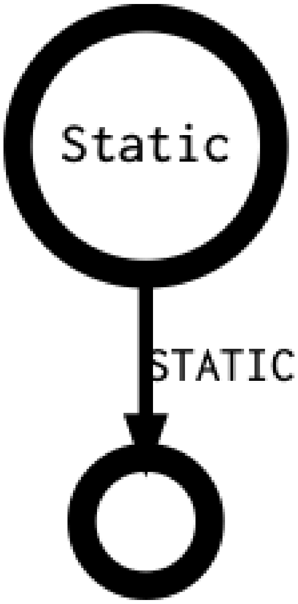
- SIGN

List of incoming links :

- ARGUMENT
- CAST
- CODE
- ELSE
- LEFT
- RETURN
- RIGHT
- THEN
- VALUE

Static

The static keyword, when it is used to refer to the current class.



List of available properties :

- code
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- rank

List of possible tokens :

- T_STATIC

List of outgoing links :

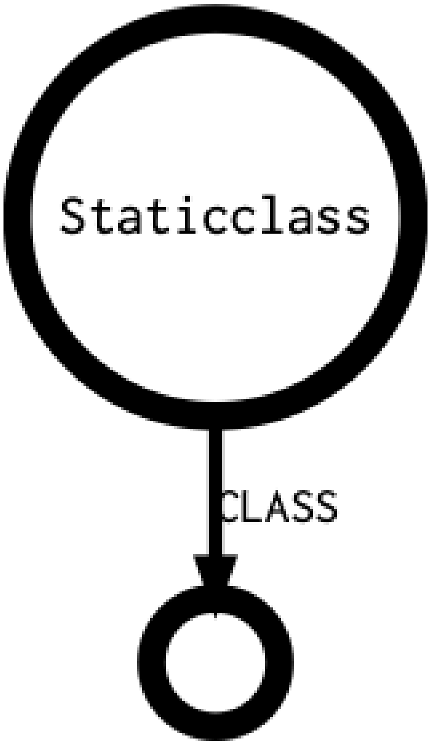
- STATIC

List of incoming links :

- CLASS
- DEFINITION
- EXPRESSION
- NAME

Staticclass

A call to *::class*, with the syntax of a static constant. For example, *X::class*.



List of available properties :

- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- intval
- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- T_DOUBLE_COLON

List of outgoing links :

- CLASS

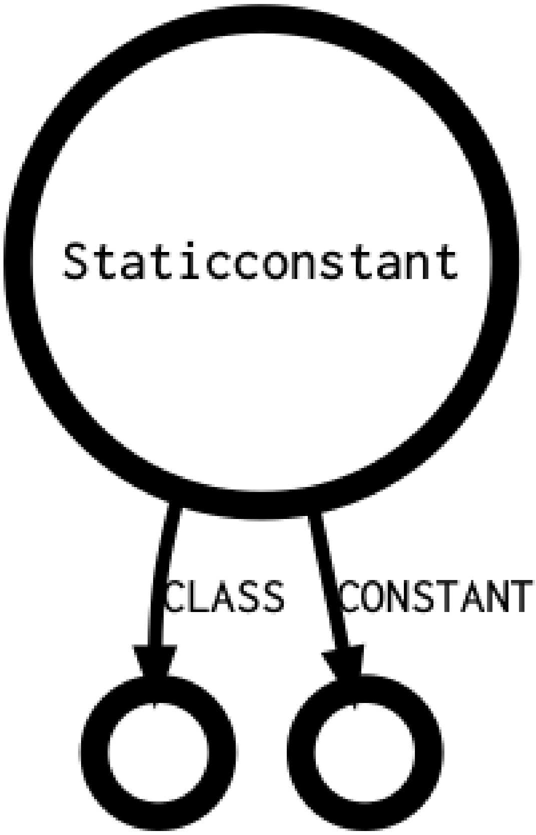
List of incoming links :

- ANALYZED
- ARGUMENT
- CASE

- CODE
- CONCAT
- DEFAULT
- ELSE
- INDEX
- LEFT
- RETURN
- RIGHT
- THEN
- VALUE
- YIELD

Staticconstant

A staticconstant : *TheClass::TheConstant*



List of available properties :

- boolean
- code
- constant

- ctype1
- ctype1_size
- fullcode
- intval
- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- T_DOUBLE_COLON
- 1

List of outgoing links :

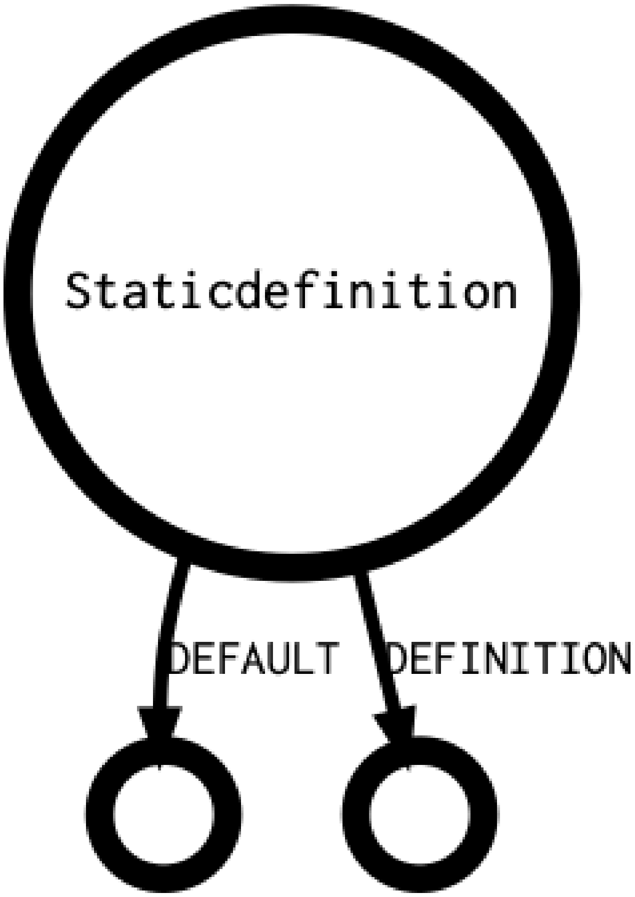
- CLASS
- CONSTANT

List of incoming links :

- ANALYZED
- ARGUMENT
- CASE
- CODE
- CONCAT
- CONDITION
- DEFAULT
- DEFINITION
- ELSE
- EXPRESSION
- INDEX
- LEFT
- NOT
- RETURN
- RIGHT
- THEN
- VALUE
- VARIABLE

Staticdefinition

A static variable definition, in a method or function. This is not a static property. For example ; *function foo() { static $s; }*.



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_VARIABLE

List of outgoing links :

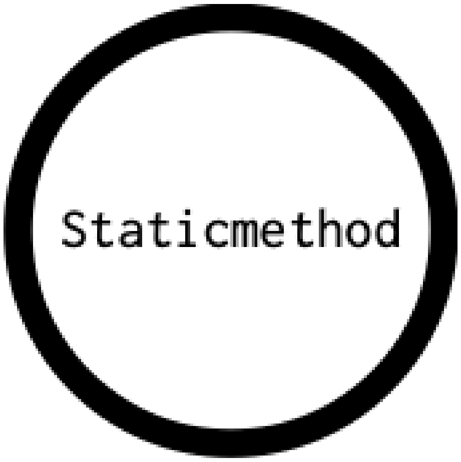
- DEFAULT
- DEFINITION

List of incoming links :

- ANALYZED
- DEFINITION
- GLOBAL
- STATIC

Staticmethod

A staticmethod name, when using trait and renaming a method. For example, *trait t { use t2 { C::D as E; }}*. *C::D* is a static method.



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line

List of possible tokens :

- T_DOUBLE_COLON
- T_STRING

List of outgoing links :

- CLASS
- METHOD

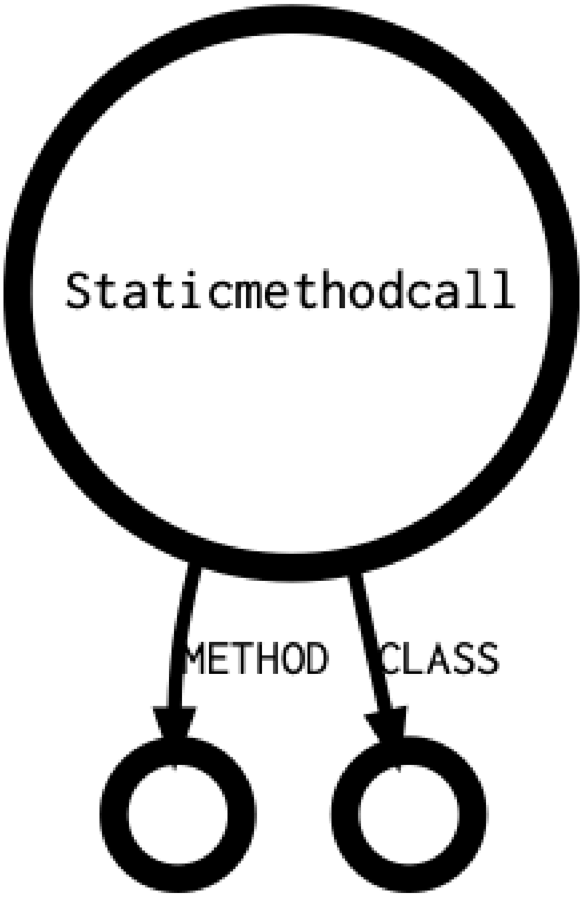
List of incoming links :

- ANALYZED
- DEFINITION
- NAME

- NEW
- SOURCE

Staticmethodcall

A static methodcall



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_DOUBLE_COLON

List of outgoing links :

- CLASS

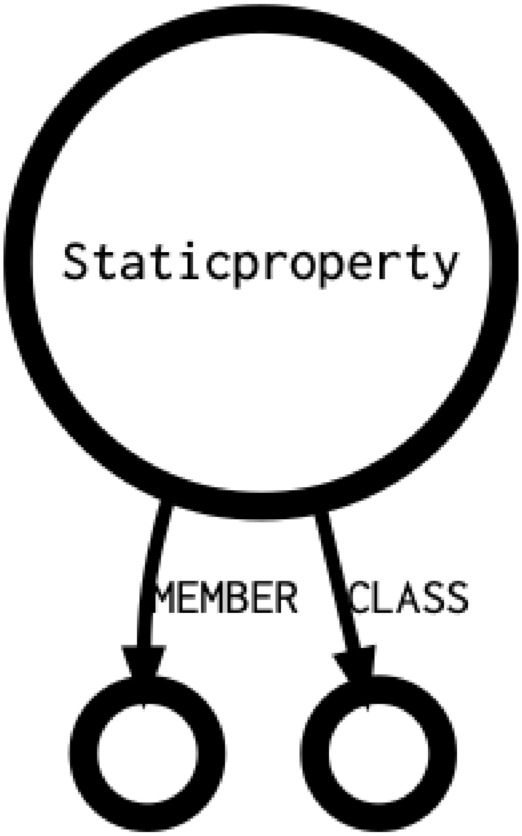
- METHOD

List of incoming links :

- ANALYZED
- APPEND
- ARGUMENT
- CAST
- CODE
- CONCAT
- CONDITION
- DEFINITION
- ELSE
- EXPRESSION
- INDEX
- LEFT
- NAME
- NEW
- NOT
- OBJECT
- RETURN
- RIGHT
- SIGN
- SOURCE
- THEN
- THROW
- VALUE
- VARIABLE

Staticproperty

A static property syntax. For example, *A::$b* or *self::$d*.



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_DOUBLE_COLON
- 1

List of outgoing links :

- CLASS
- MEMBER

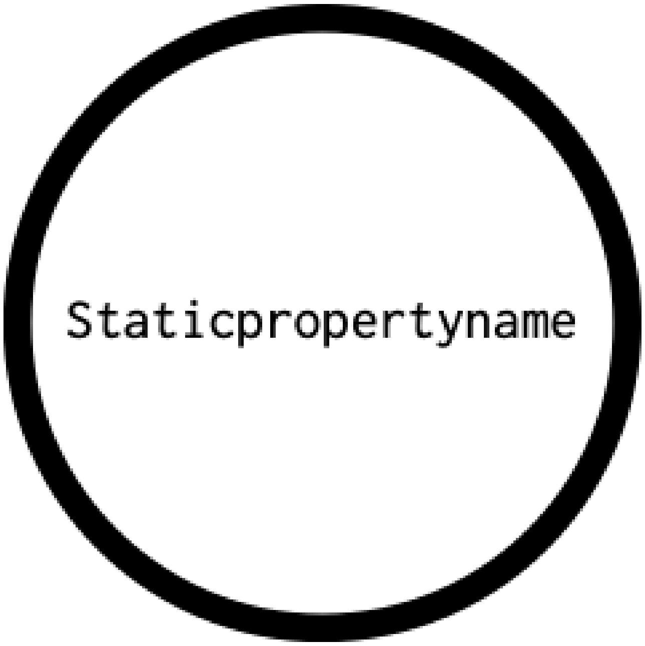
List of incoming links :

- ANALYZED
- APPEND
- ARGUMENT

- CASE
- CODE
- CONCAT
- CONDITION
- DEFINITION
- ELSE
- EXPRESSION
- INDEX
- LEFT
- NOT
- OBJECT
- POSTPLUSPLUS
- PREPLUSPLUS
- RETURN
- RIGHT
- SOURCE
- THEN
- VALUE
- VARIABLE

Staticpropertyname

The name of a static property : not a variable.



List of available properties :

- code
- ctype1_size
- fullcode
- lccode
- line

List of possible tokens :

- T_VARIABLE
- 1

List of outgoing links :

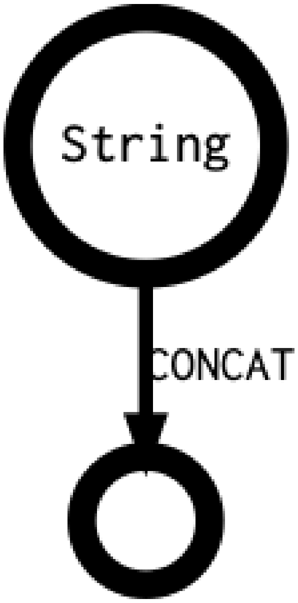
-

List of incoming links :

- ANALYZED
- DEFINITION
- MEMBER
- NAME

String

A string literal, with or without interpolation. For example, *'$x'*, *"a{$y}"*, *"a"*.



List of available properties :

- binaryString
- block
- boolean

- cbMethod
- code
- constant
- count
- ctype1
- ctype1_size
- delimiter
- encoding
- fullcode
- fullnspath
- intval
- lccode
- line
- noDelimiter
- noscream
- rank
- variadic

List of possible tokens :

- T_CONSTANT_ENCAPSED_STRING
- T_ENCAPSED_AND_WHITESPACE
- T_QUOTE
- T_STRING

List of outgoing links :

- CONCAT

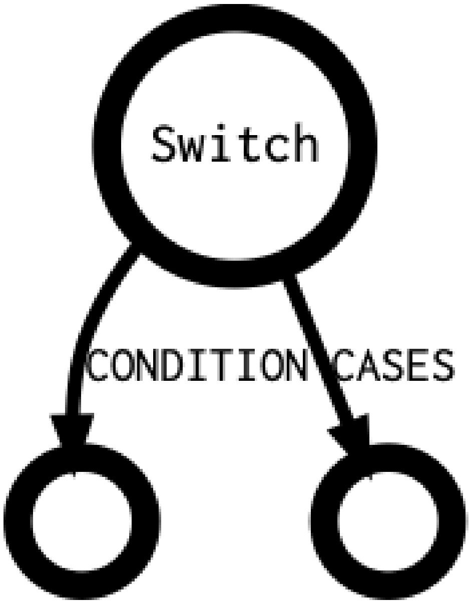
List of incoming links :

- ANALYZED
- ARGUMENT
- CASE
- CAST
- CODE
- CONCAT
- DEFAULT
- DEFINITION
- ELSE
- EXPRESSION
- INDEX

- LEFT
- NAME
- OBJECT
- RETURN
- RIGHT
- THEN
- VALUE
- VARIABLE
- YIELD

Switch

A switch structure.



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_SWITCH
- 1

List of outgoing links :

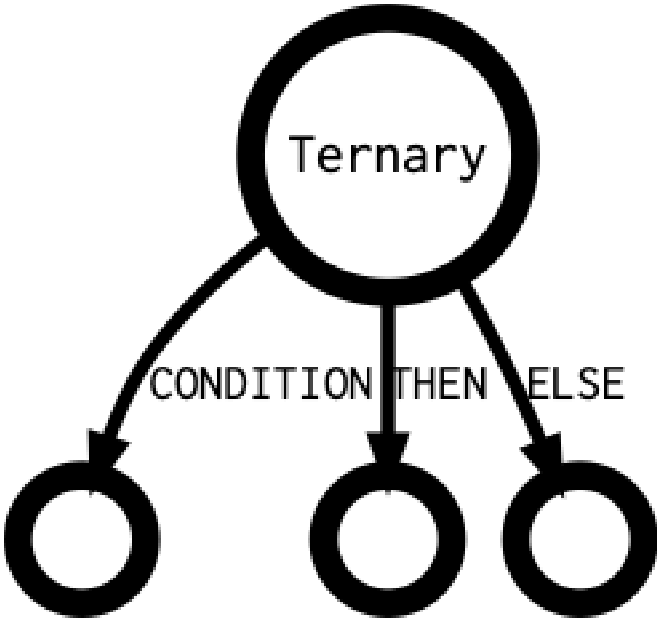
- CASES
- CONDITION

List of incoming links :

- ANALYZED
- EXPRESSION

Ternary

The ternary operator : *$a ? $b : 'c'*.



List of available properties :

- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- intval
- isNull
- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- T_QUESTION

List of outgoing links :

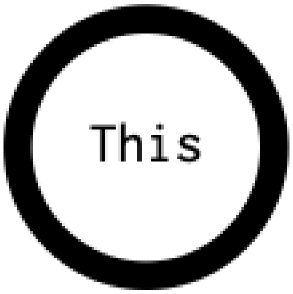
- CONDITION
- ELSE
- THEN

List of incoming links :

- ANALYZED
- ARGUMENT
- CODE
- CONDITION
- DEFAULT
- ELSE
- EXPRESSION
- INDEX
- NAME
- RETURN
- RIGHT
- SOURCE
- THEN
- THROW
- VALUE

This

The special variable *$this*.



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode

- fullnspath
- lccode
- line
- noscream
- rank
- reference

List of possible tokens :

- T_VARIABLE
- 1

List of outgoing links :

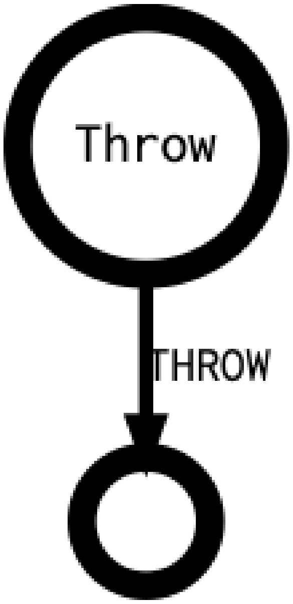
-

List of incoming links :

- ANALYZED
- ARGUMENT
- CAST
- CLASS
- CLONE
- DEFINITION
- ELSE
- EXPRESSION
- LEFT
- NEW
- OBJECT
- RETURN
- RIGHT
- SOURCE
- THEN
- VALUE
- VARIABLE
- YIELD

Throw

A throw expression



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_THROW
- 1

List of outgoing links :

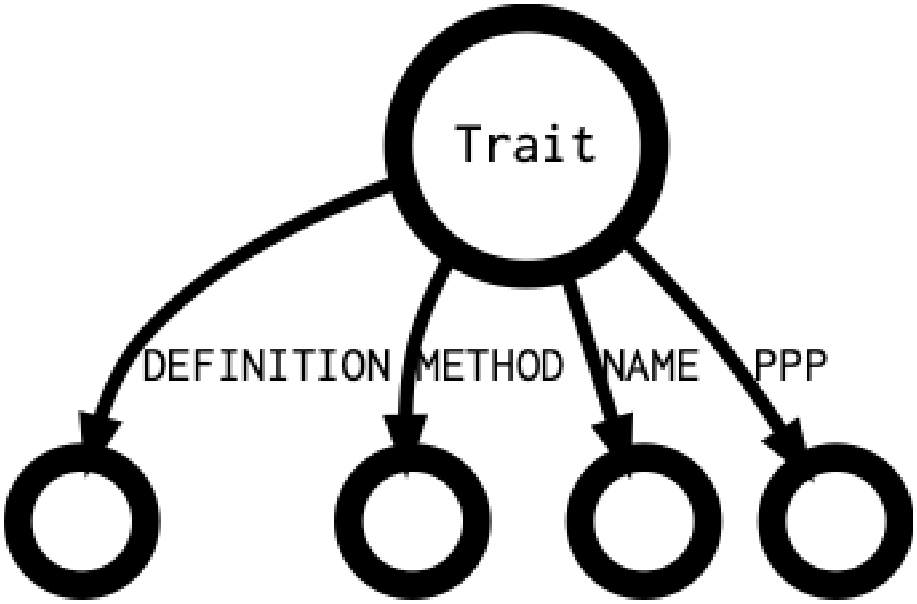
- THROW

List of incoming links :

- ANALYZED
- EXPRESSION

Trait

A trait. For example : *trait t { function foo() {} }*



List of available properties :

- code
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- rank

List of possible tokens :

- T_TRAIT

List of outgoing links :

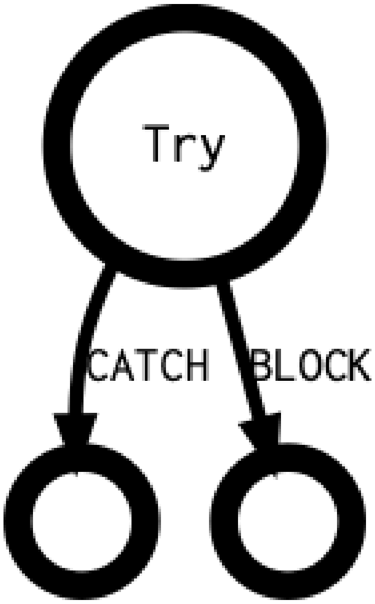
- DEFINITION
- MAGICMETHOD
- METHOD
- NAME
- PPP
- USE

List of incoming links :

- ANALYZED
- EXPRESSION

Try

The Try part in a try/catch/finally expression.



List of available properties :

- code
- count
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_TRY
- 1

List of outgoing links :

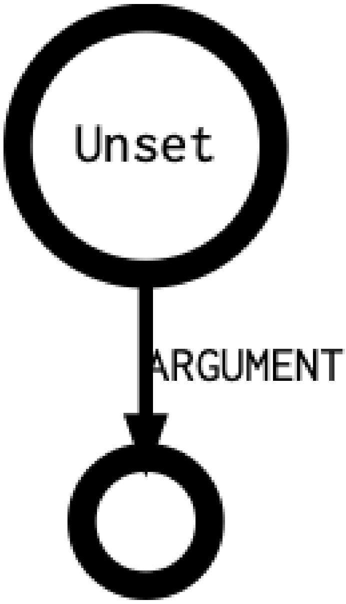
- BLOCK
- CATCH
- FINALLY

List of incoming links :

- ANALYZED
- EXPRESSION

Unset

A call to *unset*



List of available properties :

- args_max
- args_min
- code
- count
- ctype1
- ctype1_size
- fullcode
- fullnspath
- lccode
- line
- rank

List of possible tokens :

- T_UNSET
- 1

List of outgoing links :

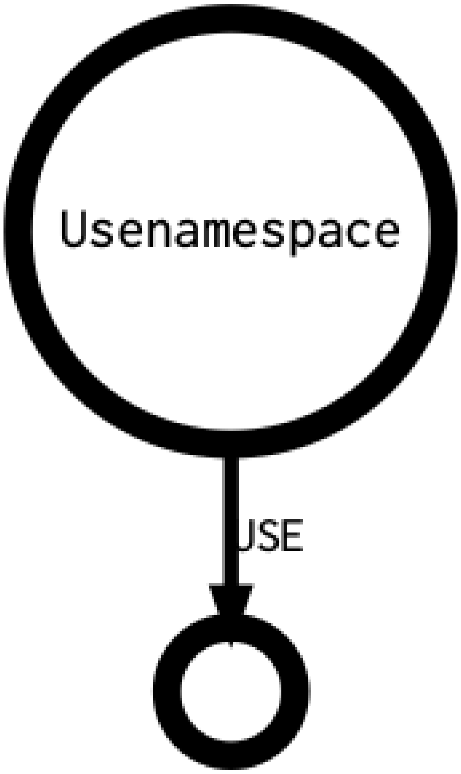
- ARGUMENT

List of incoming links :

- ANALYZED
- EXPRESSION

Usenamespace

Use expression within a namespace, and not in a class or trait.



List of available properties :

- code
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_USE
- 1

List of outgoing links :

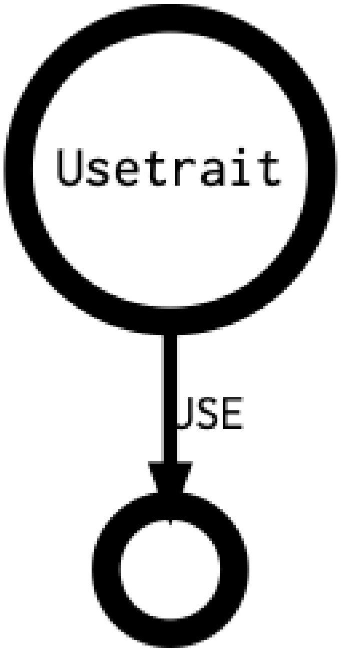
- CONST
- FUNCTION
- USE

List of incoming links :

- ANALYZED
- EXPRESSION

Usetrait

A *use* expression, when used to import a trait. For exapmle, *class x { use t; }*



List of available properties :

- code
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_USE

List of outgoing links :

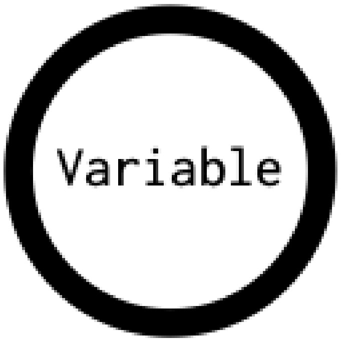
- BLOCK
- USE

List of incoming links :

- ANALYZED
- USE

Variable

A Variable, as a standalone container. For example : *$a = 1* or *$b += 3*. Variables in arrays are *Variablearray*, while variables in objects are *Variableobject*.



List of available properties :

- code
- ctype1
- ctype1_size
- enclosing
- fullcode
- fullnspath
- lccode
- line
- noDelimiter
- noscream
- rank
- reference
- variadic

List of possible tokens :

- T_CURLY_OPEN
- T_DOLLAR
- T_DOLLAR_OPEN_CURLY_BRACES
- T_STRING_VARNAME
- T_VARIABLE

List of outgoing links :

- NAME

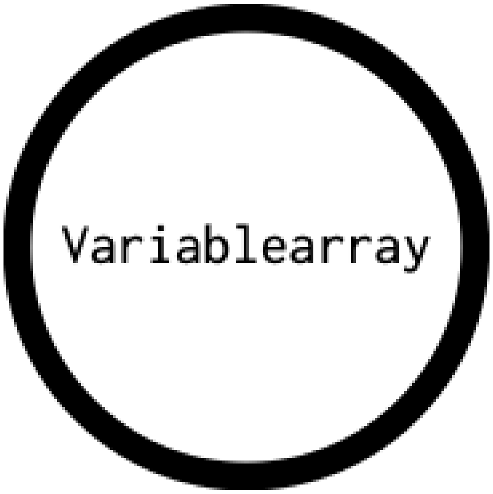
List of incoming links :

- ANALYZED
- APPEND
- ARGUMENT
- CASE
- CAST
- CLASS

- CLONE
- CODE
- CONCAT
- CONDITION
- DEFINITION
- ELSE
- EXPRESSION
- GLOBAL
- INDEX
- LEFT
- MEMBER
- NAME
- NEW
- NOT
- OBJECT
- POSTPLUSPLUS
- PREPLUSPLUS
- RETURN
- RETURNED
- RIGHT
- SIGN
- SOURCE
- THEN
- THROW
- VALUE
- VARIABLE
- YIELD

Variablearray

A variable, when used to build an array syntax. For example, the *$x* in *$x[0]* or *$x[]*.



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line

List of possible tokens :

- T_STRING_VARNAME
- T_VARIABLE

List of outgoing links :

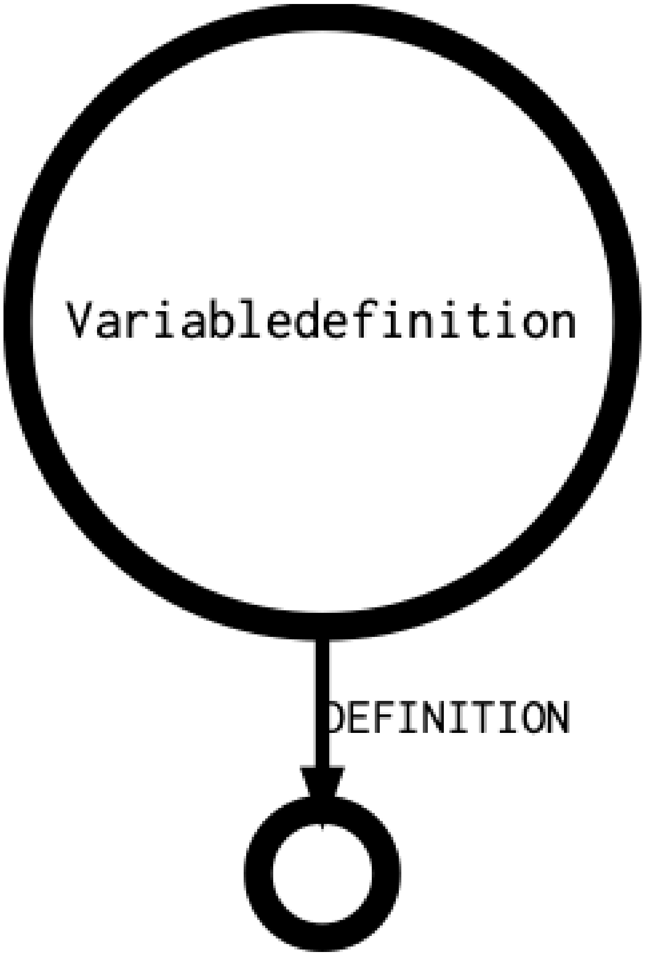
-

List of incoming links :

- ANALYZED
- APPEND
- DEFINITION
- RETURNED
- VARIABLE

Variabledefinition

A placeholder to federate local variable definition in a method.



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

-

List of outgoing links :

- DEFAULT
- DEFINITION

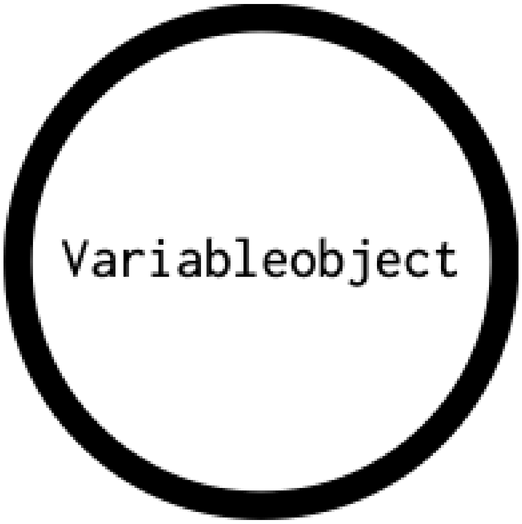
List of incoming links :

- DEFINITION
- GLOBAL

- STATIC

Variableobject

A variable when used with the -> operator.



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- noscream
- reference

List of possible tokens :

- T_VARIABLE

List of outgoing links :

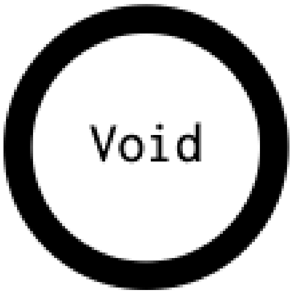
-

List of incoming links :

- ANALYZED
- DEFINITION
- NAME
- OBJECT
- RETURNED

Void

A Void operation. It represents the absence of data. For example : *foo();;* : there is a Void as argument, and one between the semicolons.



List of available properties :

- boolean
- code
- constant
- ctype1
- ctype1_size
- fullcode
- intval
- isNull
- lccode
- line
- noDelimiter
- rank

List of possible tokens :

- v

List of outgoing links :

-

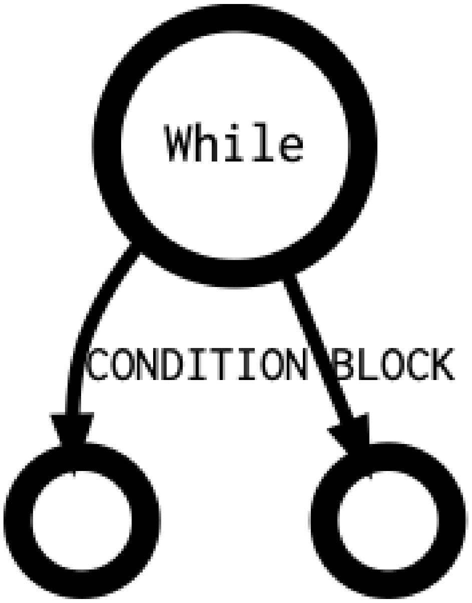
List of incoming links :

- ANALYZED
- ARGUMENT
- BLOCK
- BREAK
- CAST
- CODE
- CONTINUE
- EXPRESSION
- NAME

- RETURN
- THEN
- YIELD

While

A While structure, different from a Dowhile structure. For example : *while($a < 10) { $a++;}*



List of available properties :

- alternative
- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_WHILE

List of outgoing links :

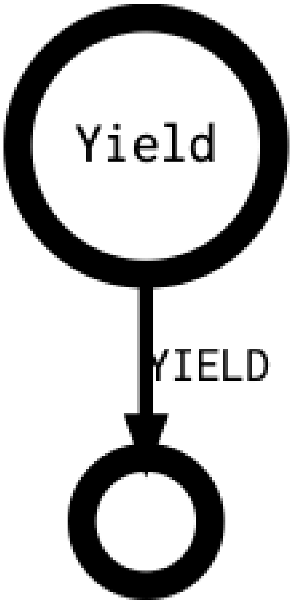
- BLOCK
- CONDITION

List of incoming links :

- ANALYZED
- EXPRESSION

Yield

A *yield* expression



List of available properties :

- code
- constant
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_YIELD

List of outgoing links :

- YIELD

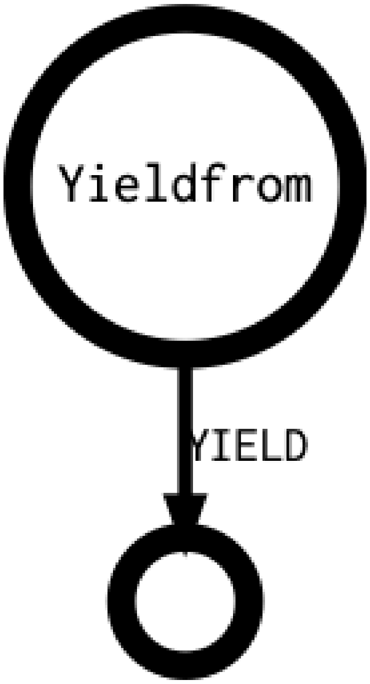
List of incoming links :

- ANALYZED
- ARGUMENT
- CONCAT
- CONDITION
- EXPRESSION
- LEFT
- RIGHT

- YIELD

Yieldfrom

A *yield from* expression



List of available properties :

- code
- ctype1
- ctype1_size
- fullcode
- lccode
- line
- rank

List of possible tokens :

- T_YIELD_FROM

List of outgoing links :

- YIELD

List of incoming links :

- ANALYZED
- ARGUMENT
- EXPRESSION
- LEFT

13.5.3 Atom properties

Each atom in the database has a list of properties. They hold information about the current atom, that are not available through the network.

To check those properties, there are some specific method calls.

- label : this is the type of atom.
- code : the value of the PHP token. For a variable, it is *$name*, while for a function, it is the function name *foo* instead of *foo(1,2,3)*
- token : the name of the current PHP token. They use the same names as inside PHP, plus a couple of special values. They are strings, and not integers
- fullcode : this is a normalized representation of the code. It include the current atom, and its important dependencies
- rank : the position of the current atom in a list of similar element, like arguments in a functioncall. rank starts at 0.
- reference : is this atom is a reference, marked with '&'
- variadic : is this atom is a variadic, marked with '. . . '
- noscream : is this atom is a variadic, marked with '@'
- block : is this atom enclosed in curly braces (only available for Sequence)
- heredoc : is this a Heredoc (true), or a Nowdoc (false) (only available for Heredoc)
- delimiter : delimiter used for string : ', " or nothing
- noDelimiter : the actual value of the string, without the delimiters
- count : count of elements. For example, count in a functioncall represents the number of arguments.
- fullnspath : the Full Qualified Name, as it was resolved at compile time.
- absolute : is this name absolute or not (only available for Nsname)
- alias : the alias name (only available for Usenamespace)
- origin : the origin for the use expression (only available for Usenamespace)
- encoding : Unicode block for the current string
- intval : the value of the atom, when cast as integer
- strval : the value of the atom, when cast as string
- boolean : the value of the atom, when cast as boolean
- args_max : maximum number of arguments (only available for Function, Method, Closure, Magicmethod)
- args_min : minimum number of arguments (only available for Function, Method, Closure, Magicmethod)
- enclosing : is the atom inside curly braces (only available for Variable inside a string)
- bracket : is the current array a short syntax or a traditional syntax (only available for Arrayliteral)
- flexible : is the Heredoc using the flexible syntax
- close_tag : has the Php atom the closing tag or not
- aliased : is the current tag aliased with a use expression, or not
- constant : is the current atom a constant value. atom are constant if they are build with constant values, like other constants or literals.

- root : is this the root node
- globalvar : the simple name of the variable, in the global syntax. For example, $GLOBALS['x'] is actually $x in the global space
- binaryString : the equivalent of strval, but after replacing the PHP escape sequence with their actual value. For example, "064" is turned into "4". This is valid for PHP sequences, unicode codepoint, etc.
- visibility : the visibility for the property, constant or method. (only available for Const, Method, Magicmethod, Propertydefinition)
- final : is the current class or method final (only available for Class, Method and Magicmethod)
- abstract : is the current class or method abstract (only available for Class, Method and Magicmethod)
- static : is the current class, property or method static (only available for Class, Method, Property and Magicmethod)

13.5.4 Links

Links are the relation established between the atoms. You can move from one to the other by using links.

Links are defined only with their label. A link between a 'Not' atom, and its operand is called 'NOT'.

There may be several links from an atom : for example, Addition has two outgoing links : 'LEFT' and 'RIGHT'.

Some links are always available, like 'CONDITION' and 'THEN' for Ifthen. 'ELSE' is not always available, depending on the code.

Some links may be repeated as often as necessary. For example, 'CONCAT' is the building block for 'Concatenation' : there may be from 2 'CONCAT' link to a lot more.

Links are oriented : they always start from the mentioned atom, and go to the next. Leaving the current atom is the 'OUT' direction, while going back to the originating atom is 'IN'.

The destination atom type is rarely defined. PHP always provides a lot of freedom, and various expressions may be used at the same place. Consider calling a function : *foo()*, *foo()*, '$foo()', *foo()()*, *$foo[1]()*. So, the target for 'NAME' from a 'Functioncall' atom, may be a 'Name', 'Nsname', 'Variable', 'Functioncall', 'Array'. Usually, it is important to always check the landing atom, before accessing properties.

13.5.5 Navigating

The script is turned into a structure network of atoms, connected by links. To create an analysis, exakat will navigate those atoms and links. The navigation is based on a specific API.

It starts with a call to atomIs() or analyzerIs() from $this, inside the analysis. Then, different steps are taken, and, in the end, if a final token is found, the query has found a result.

Initial steps

There are three special steps that must be used as first call : atomIs(), atomFunctionIs() or analyzerIs(). Those two steps are optimized as first step, to take advantage of indexes in the databases. They also represents the classic starting point of any static analysis.

Those two steps may also be used anywhere else in the query.

- atomIs($atomType) : checks that the current atom is of the type $atomType
- analyzerIs($analyzerName) : checks that the current atom is also the result of the analysis called $analyzerName.

- atomFunctionIs($functionName) : checks that the current atom is a ‘Functioncall’, with the name $functionName. This step can’t be used anywhere in the query but as the first step

Here is an example of two queries with the initial step. The first one searches for an Exit command, described above as a call to *exit* or *die*. Then, it checks that the call has no argument, which only allows ‘exit’ to be selected.

```
$this->atomIs('Exit')
    ->hasNoOut('ARGUMENT');
$this->prepareQuery();

$this->analyzerIs('Functions/IsExtFunction')
     ->outIs('ARGUMENT')
     ->atomIs('Void');
$this->prepareQuery();
```

The second is based on the ‘Functions/IsExtFunction’, which mark functioncalls made to PHP extensions : as such, the function won’t have a definition in the PHP code, but in the binary. Then, the query follows the available ‘ARGUMENT’ links, and check if the argument is ‘Void’ or not. Here, the second call to atomIs() is not an initial step.

All steps

Here is the list of the 249 available steps :

- AddEFrom : adds a link between the current atom from the atom called (see _As())
- AddETo : adds a link between the current atom to the atom called (see _As())
- AnalyzerInside : Find occurrences of results for the analyzers mentioned as argument, inside the current atom, or its children.
- AnalyzerInsideMoreThan : Docs for AnalyzerInsideMoreThan
- AnalyzerIs : checks that the current atom satisfy the analyzer
- AnalyzerIsNot : checks that the current atom doesn’t satisfy the analyzer
- AtomFunctionIs : checks that the current atom is a Functioncall with the name
- AtomInside : searches for all atom inside the current one, by searching every outgoing links
- AtomInsideExpression : Docs for AtomInsideExpression
- AtomInsideMoreThan : Docs for AtomInsideMoreThan
- AtomInsideNoAnonymous : searches for all atom inside the current one, by searching every outgoing links, but skips anonymous code like `Closure` and `Classanonymous`
- AtomInsideNoBlock : searches for all atom inside the current one, by searching every outgoing links, but skips blocks
- AtomInsideNoDefinition : searches for all atom inside the current one, by searching every outgoing links, but skips any definition, closure, class, interface, function, etc.
- AtomInsideWithCall : Searches for a method call inside the current atom.
- AtomIs : checks that an atom has a specified name
- AtomIsNot : checks that an atom is not a specified name
- Back : moves the query to the atom called () (see _As()
- CheckTypeWithAtom : Check if the current Atom is compatible with the provided scalar type. Scalar is in the full namespace path form : ‘\int’, ‘\string’, ‘\void’, . . .

- ClassDefinition : moves the query to the classDefinition, if it exists
- CodeIs : checks that the 'code' property has a given value
- CodeIsNot : checks that the 'code' property has a value different from the given one
- CodeIsPositiveInteger : Docs for CodeIsPositiveInteger
- CodeLength : report the length of the string that represents the code
- CollectArguments : Collect all arguments names, by their 'code' property, and store them in a variable (an array), named after the passed argument. Void arguments are skipped, and the final array may be empty, in case of no arguments.
- CollectContainers : Docs for CollectContainers
- CollectExtends : Docs for CollectExtends
- CollectImplements : Docs for CollectImplements
- CollectMethods : Collect all methods names, by their lowercase 'lccode' property, and store them in a variable (an array), named after the passed argument.The final array may be empty, in case of no methods.
- CollectTraits : Collect all the used traits from the current class or anonymous class, into the 'variable'. This will be a list of traits.
- CollectTypehints : Collect all the typehints of the method, property or arguments, and store them in the provided variable name. On a Method (or equivalent), this step collect the return types; On a property or an argument, it collects the types of the property or the argument. Elsewhere, it returns an empty array.
- CollectVariables : Docs for CollectVariables
- Command : Docs for Command
- Count : Docs for Count
- CountArrayDimension : Counts the number of dimensions in that array.
- CountBy : Docs for CountBy
- DSLFactory : Docs for DSLFactory
- Dedup : Docs for Dedup
- Extending : Docs for Extending
- FetchContext : Docs for FetchContext
- Filter : Docs for Filter
- FollowAlias : Follow the tracks of the current variable.
- FollowCalls : Follow calls of the argument of a function to another function. foo($a, $b) { goo($b); } : the call to $b may be followed to goo(), while no calls may be followed by $a.
- FollowExpression : Docs for FollowExpression
- FollowParAs : Follow links while skipping parenthesis, assignations, ternary and coalese operators, as they do not provide any meaning there. This step was initially called 'Follow Parenthesis Assignations'. The links provided are followed as long as they match the provided ones in argument, or the 4 atoms mentioned previously. Ternary and Coalesce are followed in all their branches.
- FollowAlias : Find all variables that are using this current one as assignement.
- FullcodeInside : Docs for FullcodeInside
- FullcodeIs : Docs for FullcodeIs

- FullcodeIsNot : Checks that the current atom's fullcode property is not one of the provided values. One value may be provided as a string, multiple values must be provided as an array of string. The step may be case-sensitive or not, by using self::CASE_SENSITIVE or self::CASE_INSENSITIVE as the second argument (default to self::CASE_INSENSITIVE)
- FullcodeLength : Docs for FullcodeLength
- FullcodeVariableIs : Docs for FullcodeVariableIs
- FullnspathIs : Docs for FullnspathIs
- FullnspathIsNot : Docs for FullnspathIsNot
- FunctionDefinition : Docs for FunctionDefinition
- FunctionInside : Docs for FunctionInside
- FunctioncallIs : Docs for FunctioncallIs
- FunctioncallIsNot : Docs for FunctioncallIsNot
- GetNameInFNP : Docs for GetNameInFNP
- GetStringLength : Docs for GetStringLength
- GetVariable : Returns the requested query variables. Those variables are initialized with initVariable.
- GoToAllChildren : Docs for GoToAllChildren
- GoToAllElse : Docs for GoToAllElse
- GoToAllImplements : Docs for GoToAllImplements
- GoToAllParents : Docs for GoToAllParents
- GoToAllParentsTraits : Docs for GoToAllParentsTraits
- GoToAllRight : Follow all the operands in a serie of chained Atoms with LEFT and RIGHT branches. This is convenient for long additions, or logical combinaisons.
- GoToAllTraits : Docs for GoToAllTraits
- GoToArray : Docs for GoToArray
- GoToClass : Docs for GoToClass
- GoToClassInterface : The traversal will go from the current atom to the first class or interface it find, upward. This may be a class, an anonymous class or an interface.
- GoToClassInterfaceTrait : Move the traverser to the class, trait or interface of the current atom, if any.
- GoToClassTrait : Move the traverser to the class or trait of the current atom, if any.
- GoToCurrentScope : Docs for GoToCurrentScope
- GoToExpression : Docs for GoToExpression
- GoToExtends : Docs for GoToExtends
- GoToFile : Move the traverser to the file of the current atom.
- GoToFirstExpression : Docs for GoToFirstExpression
- GoToFunction : Docs for GoToFunction
- GoToImplements : Docs for GoToImplements
- GoToInstruction : Docs for GoToInstruction
- GoToInterface : Docs for GoToInterface

- GoToLiteralValue : Docs for GoToLiteralValue
- GoToLoop : Go from the current atom to the closest loop. A loop is a for(), foreach(), while() or do…while().
- GoToNamespace : Docs for GoToNamespace
- GoToParameterDefinition : Move the cursor from the current argument usage, to its definition, if it is defined in the current code. This analysis is compatible with PHP 8.0 named parameters and variadic arguments.
- GoToParameterUsage : Move the cursor from the current Parameter to all its usage. This analysis is compatible with PHP 8.0 named parameters and variadic arguments.
- GoToParent : Docs for GoToParent
- GoToTrait : Docs for GoToTrait
- GoToTraits : Docs for GoToTraits
- GroupCount : Docs for GroupCount
- GroupFilter : Docs for GroupFilter
- Has : checks if a property is available for the current atom
- HasAtomInside : Docs for HasAtomInside
- HasChildWithRank : Docs for HasChildWithRank
- HasChildren : Docs for HasChildren
- HasClass : Checks that the current atom is in a class, or an anonymous class.
- HasClassDefinition : Docs for HasClassDefinition
- HasClassInterface : Docs for HasClassInterface
- HasClassTrait : Docs for HasClassTrait
- HasConstantDefinition : Docs for HasConstantDefinition
- HasFunction : Docs for HasFunction
- HasFunctionDefinition : Docs for HasFunctionDefinition
- HasIfthen : Checks that the current atom is in an if/then/else structure.
- HasIn : checks if the current atom has an incoming link with a name
- HasInstruction : Docs for HasInstruction
- HasInterface : Checks that the current atom is in an interface.
- HasInterfaceDefinition : Docs for HasInterfaceDefinition
- HasLoop : This step checks that the current atom is inside a loop structure. A loop structure is a for, a foreach, a while or a do while structure.
- HasNextSibling : Docs for HasNextSibling
- HasNo : Docs for HasNo
- HasNoCatch : Checks that the current atom is inside a catch block. The block has to be in the current scope.
- HasNoChildren : Docs for HasNoChildren
- HasNoClass : Checks that the current atom is not inside a class or an anonymous class.
- HasNoClassInterface : Checks that the current atom is not inside a class (anonymous or not), or an interface.

- HasNoClassInterfaceTrait : Checks that the current atom is not inside a class (anonymous or not), an interface or a trait.
- HasNoClassTrait : Checks that the current atom is not inside a class (anonymous or not), or a trait.
- HasNoComparison : Docs for HasNoComparison
- HasNoConstantDefinition : Docs for HasNoConstantDefinition
- HasNoCountedInstruction : Docs for HasNoCountedInstruction
- HasNoDefinition : Docs for HasNoDefinition
- HasNoFunction : Docs for HasNoFunction
- HasNoFunctionDefinition : Docs for HasNoFunctionDefinition
- HasNoIfthen : Checks that the current atom is not inside a if/then/else structure.
- HasNoIn : checks if the current atom has no incoming link with a name
- HasNoInstruction : Docs for HasNoInstruction
- HasNoInterface : This step checks that the current atom is inside an interface or not.
- HasNoLoop : Checks that the current atom is not inside a loop : foreach(), while, do… while, for.
- HasNoNamedInstruction : Docs for HasNoNamedInstruction
- HasNoNextSibling : Docs for HasNoNextSibling
- HasNoOut : checks if the current atom has no outgoing link with a name
- HasNoParent : Docs for HasNoParent
- HasNoTrait : Checks that the current atom is not inside a trait.
- HasNoTryCatch : Check if the current atom is inside a try catch structure, in the current context.
- HasNoUsage : Docs for HasNoUsage
- HasNoVariadicArgument : checks if any argument uses the variadic operator
- HasOut : checks if the current atom has no outgoing link with a name
- HasParent : Docs for HasParent
- HasPropertyInside : Docs for HasPropertyInside
- HasTrait : Checks that the current atom is in an trait.
- HasTraitDefinition : Docs for HasTraitDefinition
- HasTryCatch : Check that the current atom is inside a try/catch structure. This means a try block, a catch block or a finally block.
- HasVariadicArgument : Docs for HasVariadicArgument
- Ignore : Docs for Ignore
- Implementing : Docs for Implementing
- InIs : follows the link to the parent atom
- InIsIE : follows a link if it is present, or stay put
- InIsNot : follows a link that is not the given value
- InitVariable : Docs for InitVariable
- InterfaceDefinition : Docs for InterfaceDefinition

- InterfaceLike : Is this an abstract method, from an abstract class or an interface.
- Is : checks that the property has the value
- IsArgument : checks if the current atom is an argument of a function or method call
- IsComplexExpression : Docs for IsComplexExpression
- IsEqual : Docs for IsEqual
- IsGlobalCode : Docs for IsGlobalCode
- IsHash : Docs for IsHash
- IsInCatchBlock : Docs for IsInCatchBlock
- IsLess : Docs for IsLess
- IsLiteral : checks if an atom is a literal value
- IsLocalClass : Docs for IsLocalClass
- IsLowercase : Docs for IsLowercase
- IsMissingOrNull : Checks if the current atom has no explicit default value, and that value is not null.
- IsMore : Docs for IsMore
- IsNot : checks if a property is present, and if its value is different from the given value
- IsNotArgument : checks if an atom is not the argument of a functioncall
- IsNotEmptyArray : Docs for IsNotEmptyArray
- IsNotEmptyBody : Docs for IsNotEmptyBody
- IsNotExtendingComposer : Docs for IsNotExtendingComposer
- IsNotHash : Docs for IsNotHash
- IsNotIgnored : Check if the atom is not in one of the ignored directory.
- IsNotInheritedMethod : Docs for IsNotInheritedMethod
- IsNotLiteral : Docs for IsNotLiteral
- IsNotLocalClass : Docs for IsNotLocalClass
- IsNotLowercase : Docs for IsNotLowercase
- IsNotMixedcase : Docs for IsNotMixedcase
- IsNotNullable : Checks that a typehint doesn't include the Null type.
- IsNotPropertyDefined : Checks if the current property as no explicit definition. Exakat assign virtual definitions for every properties, when no definition has been found.
- IsNotUppercase : Docs for IsNotUppercase
- IsNullable : Checks that a typehint include the Null type.
- IsPropertyDefined : Checks if the current property as an explicit definition. Exakat assign virtual definitions for every properties, when no definition has been found.
- IsReassigned : Docs for IsReassigned
- IsReferencedArgument : Docs for IsReferencedArgument
- isThis : Checks that the current atom represent the current class. It may be `$this`, but also a property or a variable with the same type.

- IsUppercase : Docs for IsUppercase
- IsUsed : Docs for IsUsed
- IsVisible : Check if the provided visibility is compatible with the one of the atom below.
- MakeVariableName : Docs for MakeVariableName
- NextSibling : Docs for NextSibling
- NextSiblings : Docs for NextSiblings
- NoAnalyzerInside : Docs for NoAnalyzerInside
- NoAnalyzerInsideWithProperty : Docs for NoAnalyzerInsideWithProperty
- NoAtomInside : checks that the current atom has no inside its links
- NoAtomPropertyInside : Docs for NoAtomPropertyInside
- NoAtomWithoutPropertyInside : Checks that no atom, located below the current one, contains the property mentionned.
- NoChildWithRank : checks that the current atom has no children, after following the link , and checking for the rank
- NoClassDefinition : Docs for NoClassDefinition
- NoCodeInside : Docs for NoCodeInside
- NoDelimiterIs : checks that the 'noDelimiter' property has a given value
- NoDelimiterIsNot : checks that the 'noDelimiter' property has not a given value
- NoFullcodeInside : Docs for NoFullcodeInside
- NoFunctionInside : Docs for NoFunctionInside
- NoInterfaceDefinition : Docs for NoInterfaceDefinition
- NoQuery : This steps represents an empty step. It doesn't do anything, and may be used when a step is necessary, but no special process should apply.
- NoTraitDefinition : Docs for NoTraitDefinition
- NoUseDefinition : Docs for NoUseDefinition
- Not : A filter that checks that the provided sub-query doesn't return anything. If the provided sub-query returns one result, at least, then the current query stops.
- NotCompatibleWithType : Check that the provided typehint (in argument) is not compatible with the typehint or return typehint of the current parameter or method.
- NotExtending : Docs for NotExtending
- NotImplementing : Docs for NotImplementing
- NotSamePropertyAs : Docs for NotSamePropertyAs
- NotSameTypehintAs : Checks if the provided typehint is different from the current atom typehint.
- Optional : Apply the provided sub-query, only if the sub-query returns a valid value. When the subquery returns null, or fails, the current query stays in place.
- OtherSiblings : Docs for OtherSiblings
- OutIs : follow an outgoing link
- OutIsIE : follow an outgoing link if it is present, and stay put otherwise

- OutIsNot : follow an outgoing link if it is not the given value
- OutWithRank : follow an outgoing link to the given rank
- OutWithoutLastRank : Docs for OutWithoutLastRank
- PreviousCalls : Find all calls to the current methods.
- PreviousSibling : Docs for PreviousSibling
- PreviousSiblings : Docs for PreviousSiblings
- ProcessDereferencing : Count the number of dereferencing (->, (x) or [x]) in one expression. This is a dedicated analysis.
- ProcessIndentingAverage : Count the number of indentation in one expression, based on structures : if/then, switch, for, foreach, type. This is a dedicated analysis.
- ProcessLevels : Count the number of levels in one expression. This is a dedicated analysis.
- Property : Docs for Property
- PropertyIs : Docs for PropertyIs
- PropertyIsNot : Docs for PropertyIsNot
- Range : Limit the results to the values ranking from a-th to b-th.
- Raw : Runs a raw gremlin query. The query shall be a step, without any '.' before or after.
- RegexIs : apply a regex on the property
- RegexIsNot : apply a regex on the property , and checks that is fails
- ReturnCount : Docs for ReturnCount
- SamePropertyAs : Docs for SamePropertyAs
- SamePropertyAsArray : Docs for SamePropertyAsArray
- SameTypehintAs : Checks if the provided typehint is equal to the current atom typehint.
- SaveMethodNameAs : Docs for SaveMethodNameAs
- SaveOutAs : Docs for SaveOutAs
- SavePropertyAs : Docs for SavePropertyAs
- Select : Extract an array of data from the query. select() takes a array with labels as keys and properties as values. `select(array('first' => 'fullnspath'))`. This is closely related to `select` in Gremlin.
- SetProperty : Docs for SetProperty
- Side : Docs for Side
- StopQuery : StopQuery stops the current query. The query will not be executed, and will be skipped. When inside a sub-query, the sub-query will be skipped, not the main one.
- TokenIs : checks that the current atom uses the token
- TokenIsNot : checks that the current atom uses a different token than the token
- Trim : Trim the content of the provided variable with the chars in the second argument. The trim is a left trim, and the default trimmed values are single quotes and double quotes.
- Unique : Docs for Unique
- Values : Docs for Values
- VariableIsAssigned : Docs for VariableIsAssigned

- VariableIsRead : Docs for VariableIsRead
- _As : gives a unique name to the current atom. The query may come back to it with Back()

Special values

There are a few special values to be used when calling a method's query.

- Most of the arguments are expected as string. They often may also be replaced with an array of strings, and they will be used as a list of values for the same purpose. For example, *atomIs("String")* filters a "String", while *atomIs(array("String", "Integer"))* filters a "String" or an "Integer".
- With analyzerIs() and analyzerIsNot(), the special 'self' may be used to represents the current analysis.

13.5.6 Dictionaries

There are a collection of dictionaries available. Dictionaries hold list of definition, like PHP's constant and functions, extension's classes, or classes from unit test frameworks.

13.6 Documentation

Documentation is used to build automatic documentation for audit report : every time an analysis is run, its documentation is provided in the audits.

Every Exakat analysis <Folder/Name> has a documentation, stored in the 'human/en' folder, as a .ini file.

Keep the .ini files compiled, as PHP will refuse to load them otherwise. Then, Exakat will stop the processing : no documentation, no analysis.

The documentation is in international English.Localisation will be handled in the future, as other folders inside 'human'.

Each analysis has a standard structure, with the following elements :

- name : the title used for the analysis. Keep it as short as possible, as it is used for short references in reports.
- description: A complete description of the analysis. The description should include a short introduction, a detailled explanation of the targe situations, a piece of code with good recommended code as a first illustration, and various bad situations as second example. The description should also include limitations from the analysis, if any. It should also include external links, including PHP.net documentation and tutorials, to help the reader learn more about the problem.
- exakatSince: This is the version where the analysis was created. For example : "1.4.0"
- modifications: This is an array of strings : each string is a short suggestions on what kind of refactoring may be done once the analysis has spotted the issue. Suggestions should be as precise as possible. Provide as many suggestions as possible, as the problem may often be solved from different angles.

13.7 Testing your analysis

Every analysis has its own set of unit test. They check that the analysis finds every pattern it intend to find, and it doesn't find the other patterns. As such, it is important to test for expected and unwanted results.

Expected results are patterns that you expect to find. But sometimes, analysis are too broad, and collect a number extra situations that are false positives. To avoid collecting them, and to document that they should not be found, unit tests have to be written.

Analysis tests are located in the *tests/analyzer/* folder. In that folder, there is :

- Test folder : it contains the PHPUnit classes, and is automatically generated. Don't open it.
- source : this folder contains the PHP code source for the tests.
- exp : this folder contains the expected results of an analysis on the corresponding *source* code.
- random.php : this is a PHPUnit test suite that runs a random selection of unit tests
- alltests.php : this is a PHPUnit test suite that runs all the unit tests. It also checks some of the test Structures
- create_test.php : use this tool to create and add a new test to Exakat unit test list. It will create all the necessary files

Unit are run with PHPUnit version 7.0+. They were tested with PHPUnit 7.3.5 and are supposed to work with other minor versions.

13.8 Writing test

Tests must be written to match patterns and to not-match anti-patterns.

For example, imagine that we are analyzing code to find useless additions. We want to match *$a + 0*, *$a - 0*, *0 + $a* but not *0 - $a*. The last one doesn't have the same effect than the others : here the - sign has an important value. As such, *$a + 0*, *$a - 0*, *0 + $a* must go in the *$expected* array, and *0 - $a* must go in the *$expected_not* array.

The unit test framework also supports code source as folders. There are situations where PHP refuses to compile a piece of code if all the code is in a single script, but accepts the same code when split over two or several files. For that, use the *create_test.php* with *-d* option, so as to create the folder with the test. *source/Custom/MyFirst.0x.php* will be created as a folder (including with the '.php' extension). Otherwise, simply remove the *source/Custom/MyFirst.0x.php* file, and create a folder of the same name instead.

PHP source for tests only have to compile without warning. There is no need for the PHP test script to run, nor to make any sense : this code will be audited, but not run.

13.8.1 Pieces of advice

- In the PHP source for the test, always try to give names that help understand where is the error being hunter, and what are clean situations. This may be done by giving explicit names to functions and variables.
- Try to keep the PHP source in a single file. When it is not possible, rely on a directory, with little files.
- When building a test, remove any name that link it to an existing code. Often, simply changing the name '$EXPLICIT_GLOBAL' to '$X' is enough.

13.9 Tooling

There are three scripts to simplify manipulations when managing an analyzer.

They are located in the *scripts* folder. They must be called from Exakat's code root, and not from within the script folder.

- createAnalyzer <Folder/Name>: this tool creates a new analyzer in the 'Folder' folder, with the name 'Name'. At the time of creation, it creates also the documentation in 'human/en/Folder/Name.ini' file, and a first set of tests in the 'tests/analyzer/'. Finally, it sets up the analyzer in the data/analyzers.sqlite folder.
- renameAnalyzer <Folder1/Name1> <Folder2/Name2>: this tool moves the analyzer called <Folder1/Name1> to <Folder2/Name2>. It moves the code in 'library/Exakat/Analyzer/', in the tests, and in the 'human/en' folder.
- removeAnalyzer <Folder/Name>: this tool removes the analyzer called <Folder/Name>. It removes the code in 'library/Exakat/Analyzer/', in the tests, and in the 'human/en' folder.

The scripts are only available with the open source version. Exakat.phar doesn't have support for those scripts.

13.10 Publishing your analysis

To be written.

CHAPTER 14

Glossary

- *$*
    - *$_ENV*
        - *Useless Global*
    - *$_GET*
        - *Don’t Change Incomings*
        - *Indirect Injection*
        - *Useless Global*
    - *$_POST*
        - *Don’t Change Incomings*
        - *Indirect Injection*
        - *Useless Global*
    - *$_REQUEST*
        - *Indirect Injection*
        - *Useless Global*
    - *$this*
        - *$this Belongs To Classes Or Traits*
        - *$this Is Not For Static Methods*
        - *Closure May Use $this*
        - *Complex Dynamic Names*
        - *Method Could Be Static*
        - *Non Static Methods Called In A Static*
        - *Static Methods Called From Object*

    - Static Methods Can't Contain $this
    - Unbinding Closures
    - Used Once Variables
    - Using $this Outside A Class

- *
  - **
    - ** For Exponent
    - Exponent Usage
    - Mismatch Type And Default
    - Negative Power
- .
  - ...
    - Ellipsis Usage
    - Iffectations
    - Multiple Definition Of The Same Argument
    - Reserved Keywords In PHP 7
    - Should Use Operator
- @
  - @
    - @ Operator
- A
  - abs()
    - Always Positive Comparison
    - No Real Comparison
  - addslashes()
    - Filter To add_slashes()
  - array()
    - Array_merge Needs Array Of Arrays
    - Constant Scalar Expressions
    - Could Be Class Constant
    - Group Use Trailing Comma
    - Short Syntax For Arrays
    - array_merge() And Variadic
  - array_change_key_case()
    - Use Constant As Arguments
  - array_chunk()

    - *Array_merge Needs Array Of Arrays*
    - *No array_merge() In Loops*
    - *Unpacking Inside Arrays*
    - *Use Array Functions*
    - *array_merge() And Variadic*
  - *array_merge_recursive()*
    - *No array_merge() In Loops*
    - *array_merge() And Variadic*
  - *array_multisort()*
    - *Use Constant As Arguments*
  - *array_pad()*
    - *Array_Fill() With Objects*
  - *array_product()*
    - *Use Array Functions*
  - *array_push()*
    - *Avoid array_push()*
    - *Should Use Operator*
    - *Use Array Functions*
  - *array_search()*
    - *Slow Functions*
    - *Strpos()-like Comparison*
  - *array_slice()*
    - *Use Array Functions*
  - *array_splice()*
    - *Use array_slice()*
  - *array_sum()*
    - *Avoid Concat In Loop*
    - *Static Loop*
    - *Use Array Functions*
  - *array_udiff()*
    - *Slow Functions*
  - *array_uintersect()*
    - *Slow Functions*
  - *array_unique()*
    - *Avoid array_unique()*
    - *Could Use array_unique*

    - *Unconditional Break In Loop*
    - *Unreachable Code*
- ***C***
  - *Closure*
    - *Argument Should Be Typehinted*
    - *Closure Could Be A Callback*
    - *Could Be Static Closure*
    - *Unused Inherited Variable In Closure*
  - *Compact()*
    - *Compact Inexistant Variable*
    - *Could Use Compact*
  - *Count()*
    - *Can't Count Non-Countable*
    - *Uses Default Values*
  - *call_user_func()*
    - *Should Use Operator*
  - *call_user_method()*
    - *PHP 7.0 Removed Functions*
  - *call_user_method_array()*
    - *PHP 7.0 Removed Functions*
  - *chdir()*
    - *No Hardcoded Path*
  - *chmod()*
    - *Keep Files Access Restricted*
  - *chr()*
    - *Should Preprocess Chr()*
    - *Should Use Operator*
  - *chroot()*
    - *No Hardcoded Path*
  - *class_exists()*
    - *Undefined ::class*
  - *closure*
    - *Closure Could Be A Callback*
    - *Closure May Use $this*
    - *Could Be Static Closure*
    - *Parent, Static Or Self Outside Class*

    - *Should Use array_filter()*
    - *Unbinding Closures*
    - *Using $this Outside A Class*
    - *preg_replace With Option e*
  - *collator_compare()*
    - *Strpos()-like Comparison*
  - *collator_get_sort_key()*
    - *Strpos()-like Comparison*
  - *compact()*
    - *Compact Inexistant Variable*
  - *constant()*
    - *Fully Qualified Constants*
    - *PHP 7.4 Reserved Keyword*
  - *continue*
    - *Break Outside Loop*
    - *Continue Is For Loop*
    - *No Need For Else*
    - *No Return Or Throw In Finally*
    - *Unconditional Break In Loop*
    - *Unreachable Code*
  - *convert_cyr_string()*
    - *PHP 7.4 Removed Functions*
  - *count()*
    - *Always Positive Comparison*
    - *Cache Variable Outside Loop*
    - *No Count With 0*
    - *Use Constant As Arguments*
    - *Use is_countable*
    - *Useless Check*
    - *Uses Default Values*
  - *crc32()*
    - *Crc32() Might Be Negative*
  - *create_function()*
    - *PHP 7.2 Deprecations*
    - *PHP 7.2 Removed Functions*
  - *crypt()*

    - *Use password_hash()*
    - *crypt() Without Salt*
  - *curl_exec()*
    - *Strpos()-like Comparison*
  - *curl_multi_errno()*
    - *New Functions In PHP 7.1*
  - *curl_setopt()*
    - *No Weak SSL Crypto*
  - *curl_share_errno()*
    - *New Functions In PHP 7.1*
  - *curl_share_strerror()*
    - *New Functions In PHP 7.1*
  - *curl_version()*
    - *curl_version() Has No Argument*
  - *current()*
    - *Foreach Don't Change Pointer*
    - *Strpos()-like Comparison*
- ***D***
  - *DateTime*
    - *Timestamp Difference*
    - *Use DateTimeImmutable Class*
  - *DateTimeImmutable*
    - *Use DateTimeImmutable Class*
  - *Datetime*
    - *Use DateTimeImmutable Class*
  - *Die()*
    - *Print And Die*
  - *date()*
    - *Abstract Away*
  - *date_create()*
    - *PHP 7.1 Microseconds*
  - *datetime*
    - *Timestamp Difference*
    - *Use DateTimeImmutable Class*
  - *datetimeimmutable*
    - *Use DateTimeImmutable Class*

    - *Variable Is Not A Condition*
  - *ereg()*
    - *PHP 7.0 Removed Functions*
  - *ereg_replace()*
    - *PHP 7.0 Removed Functions*
  - *eregi()*
    - *PHP 7.0 Removed Functions*
  - *eregi_replace()*
    - *PHP 7.0 Removed Functions*
  - *error_clear_last()*
    - *New Functions In PHP 7.0*
  - *error_get_last()*
    - *$php_errormsg Usage*
  - *error_reporting()*
    - *Use Constant As Arguments*
  - *eval()*
    - *Eval() Usage*
    - *preg_replace With Option e*
  - *exit*
    - *Exit() Usage*
    - *Unreachable Code*
  - *exit()*
    - *Unreachable Code*
  - *explode()*
    - *Implode One Arg*
    - *Optimize Explode()*
    - *Should Use Explode Args*
  - *extract()*
    - *Configure Extract*
    - *Foreach With list()*
    - *Register Globals*
    - *Use Constant As Arguments*
  - *ezmlm_hash()*
    - *PHP 7.4 Removed Functions*
- ***F***
  - *Foreach()*

    - *Avoid array_unique()*
    - *Bracketless Blocks*
    - *Break Outside Loop*
    - *Dont Change The Blind Var*
    - *Foreach Don't Change Pointer*
    - *Foreach With list()*
    - *No Direct Usage*
    - *Overwritten Source And Value*
    - *Should Use array_column()*
    - *Should Use array_filter()*
    - *Should Yield With Key*
    - *Slow Functions*
    - *Useless Referenced Argument*
    - *preg_match_all() Flag*
  - *forward_static_call()*
    - *Callback Needs Return*
  - *forward_static_call_array()*
    - *Callback Needs Return*
  - *fputcsv()*
    - *fputcsv() In Loops*
  - *fread()*
    - *Possible Infinite Loop*
    - *Strpos()-like Comparison*
  - *fseek()*
    - *Use Constant As Arguments*
  - *func_get_arg()*
    - *func_get_arg() Modified*
  - *func_get_args()*
    - *Wrong Number Of Arguments*
    - *func_get_arg() Modified*
- ***G***
  - *gc_mem_caches()*
    - *New Functions In PHP 7.0*
  - *generator*
    - *Don't Loop On Yield*
    - *Generator Cannot Return*

        - *No Return For Generator*
    - *get_called_class()*
        - *Detect Current Class*
    - *get_class()*
        - *No Need For get_class()*
        - *No get_class() With Null*
    - *get_html_translation_table()*
        - *Use Constant As Arguments*
    - *get_magic_quotes_gpc()*
        - *PHP 7.4 Removed Functions*
    - *get_magic_quotes_runtime()*
        - *PHP 7.4 Removed Functions*
    - *glob()*
        - *Avoid glob() Usage*
        - *No Direct Usage*
    - *gmp_div_q()*
        - *Use Constant As Arguments*
    - *gmp_div_qr()*
        - *Use Constant As Arguments*
    - *gmp_div_r()*
        - *Use Constant As Arguments*
    - *gmp_random()*
        - *PHP 7.2 Deprecations*
        - *PHP 7.2 Removed Functions*
- ***H***
    - *hash()*
        - *Directly Use File*
    - *hash_equals()*
        - *Compare Hash*
    - *hash_file()*
        - *Directly Use File*
    - *hash_hmac()*
        - *Directly Use File*
    - *hash_update()*
        - *Directly Use File*
    - *hash_update_file()*

    - *Directly Use File*
  - *header()*
    - *Should Use SetCookie()*
  - *hebrevc()*
    - *PHP 7.4 Removed Functions*
  - *highlight_file()*
    - *Directly Use File*
  - *highlight_string()*
    - *Directly Use File*
  - *html_entity_decode()*
    - *Use Constant As Arguments*
  - *htmlentities()*
    - *Htmlentities Calls*
    - *Use Constant As Arguments*
    - *Uses Default Values*
  - *htmlspecialchars()*
    - *Htmlentities Calls*
    - *No ENT_IGNORE*
    - *Use Constant As Arguments*
  - *htmlspecialchars_decode()*
    - *Use Constant As Arguments*
  - *http_build_query()*
    - *Use Constant As Arguments*
    - *Use Url Query Functions*
  - *http_build_url()*
    - *Use Constant As Arguments*
  - *http_parse_cookie()*
    - *Use Constant As Arguments*
  - *http_parse_params()*
    - *Use Constant As Arguments*
  - *http_redirect()*
    - *Use Constant As Arguments*
  - *http_support()*
    - *Use Constant As Arguments*
- ***I***
  - *Isset*

    - *Isset() On The Whole Array*
- *iconv()*
    - *Substring First*
- *iconv_strpos()*
    - *Strpos()-like Comparison*
- *iconv_strrpos()*
    - *Strpos()-like Comparison*
- *idn_to_ascii()*
    - *idn_to_ascii() New Default*
- *idn_to_utf8()*
    - *idn_to_ascii() New Default*
- *image2wbmp()*
    - *PHP 7.3 Removed Functions*
    - *PHP 8.0 Removed Functions*
- *imagecolorallocate()*
    - *Strpos()-like Comparison*
- *imagecolorallocatealpha()*
    - *Strpos()-like Comparison*
- *imagepsbbox()*
    - *PHP 7.0 Removed Functions*
- *imagepsencodefont()*
    - *PHP 7.0 Removed Functions*
- *imagepsextendfont()*
    - *PHP 7.0 Removed Functions*
- *imagepsfreefont()*
    - *PHP 7.0 Removed Functions*
- *imagepsloadfont()*
    - *PHP 7.0 Removed Functions*
- *imagepsslantfont()*
    - *PHP 7.0 Removed Functions*
- *imagepstext()*
    - *PHP 7.0 Removed Functions*
- *implode()*
    - *Avoid Concat In Loop*
    - *Implode One Arg*
    - *Implode() Arguments Order*

    - *Joining file()*
    - *Use Array Functions*
- *import_request_variables()*
    - *Register Globals*
- *in_array()*
    - *Logical To in_array*
    - *Processing Collector*
    - *Slow Functions*
    - *Strict Comparison With Booleans*
- *ini_get()*
    - *PHP 8.0 Removed Directives*
- *instanceof*
    - *Should Use Operator*
    - *Undefined ::class*
    - *Unresolved Instanceof*
    - *Use Instanceof*
    - *Useless Interfaces*
- *intdiv()*
    - *Could Use Try*
    - *New Functions In PHP 7.0*
- *intval()*
    - *Should Typecast*
- *is_a()*
    - *Is_A() With String*
- *is_array()*
    - *Should Use Operator*
- *is_callable()*
    - *Check All Types*
- *is_int()*
    - *Should Use Operator*
- *is_integer()*
    - *Use Instanceof*
- *is_iterable()*
    - *Check All Types*
    - *New Functions In PHP 7.1*
- *is_null()*

    - *Use Constant As Arguments*
  - *ksort()*
    - *Use Constant As Arguments*
- ***L***
  - *List()*
    - *List With Appends*
  - *ldap_sort()*
    - *PHP 8.0 Removed Functions*
  - *list()*
    - *Empty List*
    - *Foreach With list()*
    - *List Short Syntax*
    - *List With Keys*
    - *No List With String*
    - *Overwritten Source And Value*
    - *Use List With Foreach*
    - *list() May Omit Variables*
  - *ltrim()*
    - *Substr To Trim*
- ***M***
  - *magic_quotes_runtime()*
    - *PHP 7.0 Removed Functions*
  - *mb_chr()*
    - *New Functions In PHP 7.1*
    - *New Functions In PHP 7.2*
  - *mb_encoding_aliases()*
    - *Mbstring Unknown Encoding*
  - *mb_list_encodings()*
    - *Mbstring Unknown Encoding*
  - *mb_ord()*
    - *New Functions In PHP 7.1*
    - *New Functions In PHP 7.2*
  - *mb_scrub()*
    - *New Functions In PHP 7.1*
    - *New Functions In PHP 7.2*
  - *mb_split()*

        - *Functions Removed In PHP 5.5*
        - *PHP 7.0 Removed Functions*
    - *md5()*
        - *Directly Use File*
    - *md5_file()*
        - *Directly Use File*
    - *microtime()*
        - *Use random_int()*
    - *mkdir()*
        - *Keep Files Access Restricted*
        - *Mkdir Default*
    - *money_format()*
        - *PHP 7.4 Removed Functions*
    - *move_uploaded_file()*
        - *move_uploaded_file Instead Of copy*
    - *mt_rand()*
        - *Use random_int()*
    - *mt_srand()*
        - *Use random_int()*
- *N*
    - *NULL*
        - *$this Is Not For Static Methods*
        - *Check JSON*
        - *Static Methods Can't Contain $this*
        - *Strpos()-like Comparison*
        - *array_key_exists() Speedup*
    - *Null*
        - *Avoid Optional Properties*
        - *Duplicate Literal*
        - *Indices Are Int Or String*
        - *Null Or Boolean Arrays*
        - *Scalar Or Object Property*
    - *next()*
        - *Foreach Don't Change Pointer*
        - *Static Loop*
        - *Strpos()-like Comparison*

- *PARENT*
  - *Use Lower Case For Parent, Static And Self*
- *Parent*
  - *Avoid Self In Interface*
  - *Parent, Static Or Self Outside Class*
- *pack()*
  - *Invalid Pack Format*
- *parent*
  - *Abstract Or Implements*
  - *Abstract Static Methods*
  - *Already Parents Trait*
  - *Cancel Common Method*
  - *Could Be Abstract Class*
  - *Could Be Parent Method*
  - *Could Use __DIR__*
  - *Cyclic References*
  - *Fossilized Method*
  - *Incompatible Signature Methods*
  - *Incompatible Signature Methods With Covariance*
  - *Mismatch Properties Typehints*
  - *Must Call Parent Constructor*
  - *Never Used Properties*
  - *PHP7 Dirname*
  - *Parent First*
  - *Parent, Static Or Self Outside Class*
  - *Redefined Property*
  - *Repeated Interface*
  - *Should Use Local Class*
  - *Undefined Parent*
  - *Undefined static:: Or self::*
  - *Unused Arguments*
  - *Unused Interfaces*
  - *Use Lower Case For Parent, Static And Self*
  - *self, parent, static Outside Class*
- *parse_ini_file()*
  - *Directly Use File*

        - Use Constant As Arguments
    - parse_ini_string()
        - Directly Use File
        - Use Constant As Arguments
    - parse_str()
        - PHP 7.2 Deprecations
        - Register Globals
        - Use Url Query Functions
        - parse_str() Warning
    - parse_url()
        - Pathinfo() Returns May Vary
        - Use Constant As Arguments
        - Use Url Query Functions
    - parsekit_compile_file()
        - Directly Use File
    - parsekit_compile_string()
        - Directly Use File
    - password_hash()
        - Compare Hash
        - Use password_hash()
    - password_verify()
        - Compare Hash
    - pathinfo()
        - Pathinfo() Returns May Vary
        - Use Constant As Arguments
        - Use Pathinfo
        - Use pathinfo() Arguments
    - pcntl_getpriority()
        - Strpos()-like Comparison
    - pg_result_status()
        - Use Constant As Arguments
    - pg_select()
        - Use Constant As Arguments
    - php_egg_logo_guid()
        - Functions Removed In PHP 5.5
    - php_logo_guid()

  - *Functions Removed In PHP 5.5*
- *php_real_logo_guid()*
  - *Functions Removed In PHP 5.5*
- *php_sapi_name()*
  - *Use Constant*
- *phpcredits()*
  - *Use Constant As Arguments*
- *phpinfo()*
  - *Phpinfo*
  - *Use Constant As Arguments*
- *phpversion()*
  - *Use Constant*
- *pi()*
  - *Use Constant*
- *png2wbmp()*
  - *PHP 7.2 Removed Functions*
  - *PHP 8.0 Removed Functions*
- *posix_access()*
  - *Use Constant As Arguments*
- *pow()*
  - *** For Exponent*
  - *Negative Power*
- *preg_filter()*
  - *Regex On Arrays*
- *preg_grep()*
  - *Regex On Arrays*
  - *Use Constant As Arguments*
- *preg_match()*
  - *Results May Be Missing*
  - *Strpos()-like Comparison*
  - *Use Constant As Arguments*
- *preg_match_all()*
  - *preg_match_all() Flag*
- *preg_replace()*
  - *Make One Call With Array*
  - *Possible Missing Subpattern*

    - *Strpos()-like Comparison*
  - *readfile()*
    - *Joining file()*
  - *recode()*
    - *Directly Use File*
  - *recode_file()*
    - *Directly Use File*
  - *recode_string()*
    - *Directly Use File*
  - *register_shutdown_function()*
    - *Callback Needs Return*
  - *register_tick_function()*
    - *Callback Needs Return*
  - *restore_include_path()*
    - *PHP 7.4 Removed Functions*
  - *round()*
    - *Use Constant As Arguments*
  - *rsort()*
    - *Use Constant As Arguments*
  - *rtrim()*
    - *Substr To Trim*
  - *runkit_import()*
    - *Use Constant As Arguments*
- ***S***
  - *Self*
    - *Avoid Self In Interface*
  - *Static*
    - *$this Is Not For Static Methods*
    - *Abstract Static Methods*
    - *No Reference For Static Property*
    - *Non Static Methods Called In A Static*
    - *Parent, Static Or Self Outside Class*
    - *Should Be Single Quote*
    - *Should Use Local Class*
    - *Static Loop*
    - *Static Methods Called From Object*

    - *Static Methods Can't Contain $this*
    - *Undefined static:: Or self::*
    - *Wrong Access Style to Property*
  - *Strtr()*
    - *Strtr Arguments*
  - *Substr()*
    - *Drop Substr Last Arg*
  - *Switch()*
    - *Missing Cases In Switch*
  - *scandir()*
    - *Avoid glob() Usage*
    - *Use Constant As Arguments*
  - *self*
    - *Abstract Static Methods*
    - *No Self Referencing Constant*
    - *Parent, Static Or Self Outside Class*
    - *Undefined static:: Or self::*
    - *Unused Methods*
    - *Unused Private Methods*
    - *Upload Filename Injection*
    - *Use Lower Case For Parent, Static And Self*
    - *self, parent, static Outside Class*
    - *strip_tags Skips Closed Tag*
  - *session_start()*
    - *Use session_start() Options*
  - *set_error_handler()*
    - *Avoid set_error_handler $context Argument*
  - *set_exception_handler()*
    - *set_exception_handler() Warning*
  - *set_magic_quotes_runtime()*
    - *PHP 7.0 Removed Functions*
  - *set_socket_blocking()*
    - *PHP 7.0 Removed Functions*
  - *setcookie()*
    - *Set Cookie Safe Arguments*
    - *Should Use SetCookie()*

- *setlocale()*
  - *Setlocale() Uses Constants*
- *setrawcookie()*
  - *Set Cookie Safe Arguments*
  - *Should Use SetCookie()*
- *settype()*
  - *Should Typecast*
- *sha1()*
  - *Directly Use File*
- *sha1_file()*
  - *Directly Use File*
- *show_source()*
  - *Directly Use File*
- *simplexml_load_file()*
  - *Directly Use File*
- *simplexml_load_string()*
  - *Directly Use File*
- *sizeof()*
  - *Useless Check*
- *sleep()*
  - *Avoid sleep()/usleep()*
- *socket_read()*
  - *Use Constant As Arguments*
- *sort()*
  - *Use Constant As Arguments*
- *split()*
  - *PHP 7.0 Removed Functions*
- *spliti()*
  - *PHP 7.0 Removed Functions*
- *sql_regcase()*
  - *PHP 7.0 Removed Functions*
- *srand()*
  - *Use random_int()*
- *static*
  - *$this Belongs To Classes Or Traits*
  - *$this Is Not For Static Methods*

- *Abstract Static Methods*
- *Ambiguous Static*
- *Cant Use Return Value In Write Context*
- *Closure Could Be A Callback*
- *Constant Scalar Expressions*
- *Could Be Static*
- *Could Be Static Closure*
- *Dependant Abstract Classes*
- *Dependant Trait*
- *Don't Unset Properties*
- *Forgotten Visibility*
- *Magic Visibility*
- *Method Could Be Static*
- *Mismatch Type And Default*
- *Modified Typed Parameter*
- *No Direct Call To Magic Method*
- *No Need For get_class()*
- *No Net For Xml Load*
- *No Reference For Static Property*
- *No Return Used*
- *Non Static Methods Called In A Static*
- *Only Variable For Reference*
- *Only Variable Passed By Reference*
- *Only Variable Returned By Reference*
- *Order Of Declaration*
- *Parent, Static Or Self Outside Class*
- *Property Used In One Method Only*
- *Should Use Local Class*
- *Static Global Variables Confusion*
- *Static Loop*
- *Static Methods Called From Object*
- *Static Methods Can't Contain $this*
- *Too Many Dereferencing*
- *Too Many Local Variables*
- *Unbinding Closures*
- *Undefined Variable*

  - *Undefined static:: Or self::*
  - *Unused Private Properties*
  - *Use Class Operator*
  - *Use Lower Case For Parent, Static And Self*
  - *Use PHP7 Encapsed Strings*
  - *Used Once Variables*
  - *Useless Abstract Class*
  - *Useless Unset*
  - *Using $this Outside A Class*
  - *Wrong Access Style to Property*
  - *self, parent, static Outside Class*
- *str_ireplace()*
  - *Make One Call With Array*
- *str_pad()*
  - *Could Use str_repeat()*
  - *Use Constant As Arguments*
- *str_repeat()*
  - *Could Use str_repeat()*
- *str_replace()*
  - *Joining file()*
  - *Make One Call With Array*
- *stream_isatty()*
  - *New Functions In PHP 7.2*
- *stream_socket_client()*
  - *Use Constant As Arguments*
- *stream_socket_server()*
  - *@ Operator*
  - *Use Constant As Arguments*
- *strip_tags()*
  - *strip_tags Skips Closed Tag*
- *stripos()*
  - *Simplify Regex*
  - *Strpos()-like Comparison*
  - *strpos() Too Much*
- *strlen()*
  - *Always Positive Comparison*

    - *No Count With 0*
  - *strpos()*
    - *Simplify Regex*
    - *Slow Functions*
    - *Strpos()-like Comparison*
    - *strpos() Too Much*
  - *strripos()*
    - *Strpos()-like Comparison*
  - *strrpos()*
    - *Strpos()-like Comparison*
  - *strstr()*
    - *Slow Functions*
  - *strtok()*
    - *Strpos()-like Comparison*
  - *strtotime()*
    - *Next Month Trap*
    - *time() Vs strtotime()*
  - *strtr()*
    - *Strtr Arguments*
  - *strval()*
    - *Concat Empty String*
  - *substr()*
    - *Avoid Substr() One*
    - *No List With String*
    - *No mb_substr In Loop*
    - *Substr To Trim*
    - *Substring First*
    - *Use Basename Suffix*
    - *Use array_slice()*
    - *strpos() Too Much*
  - *substr_replace()*
    - *Make One Call With Array*
  - *switch()*
    - *Bracketless Blocks*
    - *Break Outside Loop*
    - *Missing Cases In Switch*

    - *Strict Comparison With Booleans*
    - *Switch To Switch*
    - *Switch With Too Many Default*
    - *Switch Without Default*
    - *Use Case Value*
  - *sys_get_temp_dir()*
    - *No Hardcoded Path*
    - *Use System Tmp*
- ***T***
  - *Throwable*
    - *Can't Throw Throwable*
    - *Empty Try Catch*
  - *throwable*
    - *Can't Throw Throwable*
  - *time()*
    - *Use random_int()*
    - *time() Vs strtotime()*
  - *token_get_all()*
    - *@ Operator*
  - *traversable*
    - *Cant Implement Traversable*
  - *trigger_error()*
    - *Use Constant As Arguments*
  - *trim()*
    - *Substr To Trim*
    - *Substring First*
- ***U***
  - *Unset()*
    - *Multiple Unset()*
  - *Usort()*
    - *Usort Sorting In PHP 7.0*
  - *uasort()*
    - *Slow Functions*
    - *Usort Sorting In PHP 7.0*
  - *uksort()*
    - *Slow Functions*

- __CLASS__
  - ::class
  - Detect Current Class
- __DIR__
  - Could Use __DIR__
  - No Hardcoded Path
  - __DIR__ Then Slash
- __FILE__
  - Could Use __DIR__
  - No Hardcoded Path
- __call
  - Check On __Call Usage
  - Must Return Methods
  - No Direct Call To Magic Method
  - Useless Typehint
- __callStatic
  - Must Return Methods
- __clone
  - Direct Call To __clone()
  - No Direct Call To Magic Method
  - Should Deep Clone
- __construct
  - Cant Instantiate Class
  - Could Use Promoted Properties
  - Don't Send $this In Constructor
  - Must Call Parent Constructor
  - Old Style Constructor
  - Should Chain Exception
- __debugInfo
  - Must Return Methods
  - __debugInfo() Usage
- __get
  - Must Return Methods
  - No Direct Call To Magic Method
  - Useless Typehint
- __invoke

    - *Must Return Methods*
  - *__isset*
    - *Must Return Methods*
  - *__set*
    - *Useless Typehint*
  - *__set_state*
    - *Must Return Methods*
  - *__sleep*
    - *Must Return Methods*
  - *__toString*
    - *Could Be Stringable*
    - *Must Return Methods*
    - *No Direct Call To Magic Method*
    - *Reflection Export() Is Deprecated*
    - *__toString() Throws Exception*

CHAPTER 15

Definitions

Here is a list of words, commonly used when using Exakat, with their definitions and their synonyms.

- ***A***

  Analysis An *Analysis* is a pattern that may be detected in the code. The analysis has a human-redable description, and a specific implementation.

- ***D***

  Dump The phase of execution, which prepare the results from the graph database to the data storage for reports.

- ***I***

  Issue The result of an analysis, when an analysis is applied to a code.

- ***L***

  Load The phase of execution, which loads the source code into the central database.

- ***R***

  Report A set of issues, gathered into a consistent format, after running the analysis on the code. A report may include multiple rulesets, and use various format, such as HTML, JSON or Text.

  Rule A synonym for Analysis. This may be more descriptive, and less related to implementation.

  Ruleset A consistent group of analysis, recognizable with a specific name.

CHAPTER 16

Ideas

Exakat is an Open Source project. It is also organized to collect common knowledge and encode it in its databases.

Here are some suggestions of help you may provide to enhance your own usage of Exakat :

- Suggest PHP extensions that are missing in the list of supported extensions (see *Annex*)
- Suggest new analysis, with examples of target code, and examples of good code
- Suggest missing external services
- Suggest reference article for the documentation, in the section 'See also'
- Suggestion application that may be added to the corpus of codes that we use to validate the analysis
- Provide new names and adjectives for the audit names. We like to include any first name of community members, and non-derogatory adjectives.
- Report installation or usage problems
- Report ambiguity in reports and their documentation
- Suggest interesting Coding reference, like Object Calisthenics, PSR, East-Oriented Programming, etc.
- Translate the documentation into other languages
- Suport Exakat on Windows or other OS
- Recommend article for code conception to be added in the docs
- Suggest public code source for review

Visit us on the [github repository](https://github.com/exakat/php-static-analysis-tools), or the [slack channel](https://www.exakat.io/slack-invitation/).

CHAPTER 17

List of contributors

The following people helped in the making of Exakat : installing, coding, suggesting, using, documenting, reporting bugs, pushing us to be better.

- (Buck / Leon)
- (Jent / Jean)
- Gérard Ernaelsten
- Philippe Gamache
- Cyrille Granval
- Eshin Kunishima
- Alexis Van Glasow

CHAPTER 18

Annex

- Supported Rulesets
- Supported Reports
- Supported PHP Extensions
- Supported Frameworks
- Applications
- Recognized Libraries
- New analyzers
- External services
- PHP Error messages

18.1 Supported Rulesets

Exakat groups analysis by rulesets. This way, analyzing ‘Security’ runs all possible analysis related to security. One analysis may belong to multiple rulesets.

- All
- Analyze
- Appcontent
- Appinfo
- CI-checks
- Calisthenics
- ClassReview
- ClearPHP

- Coding Conventions
- CompatibilityPHP53
- CompatibilityPHP54
- CompatibilityPHP55
- CompatibilityPHP56
- CompatibilityPHP70
- CompatibilityPHP71
- CompatibilityPHP72
- CompatibilityPHP73
- CompatibilityPHP74
- CompatibilityPHP80
- Complete
- Custom
- Dead code
- DefensiveProgrammingTM
- Dismell
- Dump
- First
- Internal
- Inventory
- Level 1
- Level 2
- Level 3
- Level 4
- Level 5
- LintButWontExec
- Newfeatures
- None
- OneFile
- PHP recommendations
- Performances
- Portability
- Preferences
- RadwellCodes
- Rector
- SOLID

- Security
- Semantics
- Simple
- Stats
- Suggestions
- Top10
- Typechecks
- Typehints
- Unassigned
- Under Work
- php-cs-fixable

18.2 Supported Reports

Exakat produces various reports. Some are general, covering various aspects in a reference way; others focus on one aspect.

- Ambassador
- Ambassadornomenu
- Drillinstructor
- Top10
- Text
- Xml
- Uml
- Yaml
- Plantuml
- None
- Simplehtml
- Owasp
- Perfile
- Beautycanon
- Phpconfiguration
- Phpcompilation
- Favorites
- Manual
- Inventories
- Clustergrammer
- Filedependencies

- Filedependencieshtml
- Classdependencies
- Stubs
- StubsJson
- Radwellcode
- Grade
- Weekly
- Scrutinizer
- Codesniffer
- Phpcsfixer
- Facetedjson
- Json
- Onepagejson
- Marmelab
- Simpletable
- Exakatyaml
- Codeflower
- Dependencywheel
- Phpcity
- Sarb
- Exakatvendors
- Topology
- Migration73
- Migration74
- Migration80
- Meters

18.3 Supported PHP Extensions

PHP extensions are used to check for structures usage (classes, interfaces, etc.), to identify dependencies and directives.

PHP extensions are described with the list of structures they define : functions, classes, constants, traits, variables, interfaces, namespaces, and directives.

- ext/amqp
- ext/apache
- ext/apc
- ext/apcu
- ext/array

- ext/php-ast
- ext/async
- ext/bcmath
- ext/bzip2
- ext/cairo
- ext/calendar
- ext/cmark
- ext/com
- ext/crypto
- ext/csprng
- ext/ctype
- ext/curl
- ext/cyrus
- ext/date
- ext/db2
- ext/dba
- ext/decimal
- ext/dio
- ext/dom
- ext/ds
- ext/eaccelerator
- ext/eio
- ext/enchant
- ext/ereg
- ext/ev
- ext/event
- ext/exif
- ext/expect
- ext/fam
- ext/fann
- ext/fdf
- ext/ffi
- ext/ffmpeg
- ext/file
- ext/fileinfo
- ext/filter

- ext/fpm
- ext/ftp
- ext/gd
- ext/gearman
- ext/gender
- ext/geoip
- ext/gettext
- ext/gmagick
- ext/gmp
- ext/gnupgp
- ext/grpc
- ext/hash
- ext/hrtime
- ext/pecl_http
- ext/ibase
- ext/iconv
- ext/igbinary
- ext/iis
- ext/imagick
- ext/imap
- ext/info
- ext/inotify
- ext/intl
- ext/json
- ext/judy
- ext/kdm5
- ext/lapack
- ext/ldap
- ext/leveldb
- ext/libevent
- ext/libsodium
- ext/libxml
- ext/lua
- ext/lzf
- ext/mail
- ext/mailparse

- ext/math
- ext/mbstring
- ext/mcrypt
- ext/memcache
- ext/memcached
- ext/mhash
- ext/ming
- ext/mongo
- ext/mongodb
- ext/msgpack
- ext/mssql
- ext/mysql
- ext/mysqli
- ext/ncurses
- ext/newt
- ext/nsapi
- ext/ob
- ext/oci8
- ext/odbc
- ext/opcache
- ext/opencensus
- ext/openssl
- ext/parle
- ext/parsekit
- ext/password
- ext/pcntl
- ext/pcov
- ext/pcre
- ext/pdo
- ext/pgsql
- ext/phalcon
- ext/phar
- ext/posix
- ext/proctitle
- ext/pspell
- ext/psr

- ext/rar
- ext/rdkafka
- ext/readline
- ext/recode
- ext/redis
- ext/reflection
- ext/runkit
- ext/sdl
- ext/seaslog
- ext/sem
- ext/session
- ext/shmop
- ext/simplexml
- ext/snmp
- ext/soap
- ext/sockets
- ext/sphinx
- ext/spl
- ext/sqlite
- ext/sqlite3
- ext/sqlsrv
- ext/ssh2
- ext/standard
- ext/stats
- String
- ext/suhosin
- ext/svm
- ext/swoole
- ext/tidy
- ext/tokenizer
- ext/tokyotyrant
- ext/trader
- ext/uopz
- ext/uuid
- ext/v8js
- ext/varnish

- ext/vips
- ext/wasm
- ext/wddx
- ext/weakref
- ext/wikidiff2
- ext/wincache
- ext/xattr
- ext/xcache
- ext/xdebug
- ext/xdiff
- ext/xhprof
- ext/xml
- ext/xmlreader
- ext/xmlrpc
- ext/xmlwriter
- ext/xsl
- ext/xxtea
- ext/yaml
- ext/yis
- ext/zbarcode
- ext/zend_monitor
- ext/zip
- ext/zlib
- ext/0mq
- ext/zookeeper

18.4 Supported Frameworks

Frameworks, components and libraries are supported via Exakat extensions.

List of extensions : there are 10 extensions

- Cakephp
- Drupal
- Laravel
- Pmb
- Prestashop
- Shopware

- Slim
- Symfony
- Wordpress
- ZendF

18.5 Applications

A number of applications were scanned in order to find real life examples of patterns. They are listed here :

- ChurchCRM
- Cleverstyle
- Contao
- Dolibarr
- Dolphin
- Edusoho
- ExpressionEngine
- FuelCMS
- HuMo-Gen
- LiveZilla
- Magento
- Mautic
- MediaWiki
- NextCloud
- OpenConf
- OpenEMR
- Phinx
- PhpIPAM
- Phpdocumentor
- Piwigo
- PrestaShop
- SPIP
- SugarCrm
- SuiteCrm
- TeamPass
- Thelia
- ThinkPHP
- Tikiwiki

- Tine20
- Traq
- Typo3
- Vanilla
- Woocommerce
- WordPress
- XOOPS
- Zencart
- Zend-Config
- Zurmo
- opencfp
- phpMyAdmin
- phpadsnew
- shopware
- xataface

18.6 Recognized Libraries

Libraries that are popular, large and often included in repositories are identified early in the analysis process, and ignored. This prevents Exakat to analysis some code foreign to the current repository : it prevents false positives from this code, and make the analysis much lighter. The whole process is entirely automatic.

Those libraries, or even some of the, may be included again in the analysis by commenting the ignored_dir[] line, in the projects/<project>/config.ini file.

- ADOdb
- atoum
- BBQ
- CakePHP
- CI xmlRPC
- CPDF
- Codeception
- DomPDF
- FPDF
- phpGACL
- gettext Reader
- jpGraph
- HTML2PDF
- HTML Purifier

- http_class
- IDNA convert
- lessc
- magpieRSS
- MarkDown Parser
- Markdown
- mpdf
- oauthToken
- passwordHash
- pChart
- pclZip
- Propel
- phpExecl
- phpMailer
- PHPSpec
- PHPUnit
- qrCode
- Services_JSON
- sfYaml
- SimplePie
- SimpleTest
- swift
- Smarty
- Symfony Unit Test
- tcpdf
- text_diff
- text highlighter
- tfpdf
- Typo3TestingFramework
- UTF8
- Xajax
- Yii
- Zend Framework

18.7 New analyzers

List of analyzers, by version of introduction, newest to oldest. In parenthesis, the first element is the analyzer name, used with 'analyze -P' command, and the seconds, if any, are the ruleset, used with the -T option. Rulesets are separated by commas, as the same analysis may be used in several rulesets.

- 2.1.9
  - Array_Fill() With Objects (Structures/ArrayFillWithObjects ; Analyze)
  - Assumptions (Php/Assumptions ; Analyze)
  - Complete/PhpExtStubPropertyMethod (Complete/PhpExtStubPropertyMethod ; Complete)
  - Could Be Stringable (Classes/CouldBeStringable ; Analyze, LintButWontExec)
  - Could Use Promoted Properties (Php/CouldUsePromotedProperties ; Suggestions)
  - Dump/CollectUseCOunts (Dump/CollectUseCounts ; Dump)
  - Modified Typed Parameter (Functions/ModifyTypedParameter ; Analyze, ClassReview)
  - Negative Start Index In Array (Arrays/NegativeStart)
  - Nullable With Constant (Functions/NullableWithConstant ; CompatibilityPHP80)
  - Optimize Explode() (Performances/OptimizeExplode ; Performances)
  - PHP 8.0 Removed Directives (Php/Php80RemovedDirective ; CompatibilityPHP80)
  - Unsupported Types With Operators (Structures/UnsupportedTypesWithOperators ; Analyze, CompatibilityPHP80)
  - Useless Typehint (Classes/UselessTypehint ; Suggestions, ClassReview)
- 2.1.8
  - $php_errormsg Usage (Php/PhpErrorMsgUsage ; CompatibilityPHP80)
  - Cancel Common Method (Classes/CancelCommonMethod)
  - Cast Unset Usage (Php/CastUnsetUsage ; CompatibilityPHP80)
  - Collect Atom Counts (Dump/CollectAtomCounts ; Dump)
  - Collect Classes Dependencies (Dump/CollectClassesDependencies ; Dump)
  - Collect Files Dependencies (Dump/CollectFilesDependencies ; Dump)
  - Collect Php Structures (Dump/CollectPhpStructures ; Dump)
  - Function With Dynamic Code (Functions/DynamicCode ; Internal)
  - Mismatch Parameter And Type (Functions/MismatchParameterAndType ; Analyze, Semantics)
  - Mismatch Parameter Name (Functions/MismatchParameterName ; Analyze, CompatibilityPHP80)
  - Multiple Declaration Of Strict_types (Php/MultipleDeclareStrict ; Analyze)
- 2.1.7
  - Collect Class Traits Counts (Dump/CollectClassTraitsCounts ; Dump)
  - Collect Native Calls Per Expressions (Dump/CollectNativeCallsPerExpressions ; Dump)
  - Collect Readability (Dump/CollectReadability ; Dump)
  - Collect Variables (Dump/CollectVariables ; Dump)

  - Could Be Parent Method (Classes/CouldBeParentMethod)
  - Don't Pollute Global Space (Php/DontPolluteGlobalSpace ; Analyze)
  - Dump/CollectDefinitionsStats (Dump/CollectDefinitionsStats ; Dump)
  - Dump/CollectGlobalVariables (Dump/CollectGlobalVariables ; Dump)
  - Dump/CollectNamespaces (Dump/CollectNamespaces ; Unassigned)
  - Missing Returntype In Method (Typehints/MissingReturntype ; Analyze, Typehints, CI-checks)
- 2.1.6
  - Different Argument Counts (Classes/DifferentArgumentCounts)
  - GLOB_BRACE Usage (Portability/GlobBraceUsage ; Portability)
  - Iconv With Translit (Portability/IconvTranslit ; Portability)
  - Unknown Parameter Name (Functions/UnknownParameterName ; Analyze, CI-checks)
  - Use Closure Trailing Comma (Php/UseTrailingUseComma ; Appinfo)
  - Use NullSafe Operator (Php/UseNullSafeOperator ; Appinfo)
  - Use PHP Attributes (Php/UseAttributes ; Appinfo)
- 2.1.5
  - Abstract Away (Patterns/AbstractAway ; Suggestions)
  - Catch Undefined Variable (Exceptions/CatchUndefinedVariable ; Analyze)
  - Collect Parameter Names (Dump/CollectParameterNames ; Dump)
  - Dont Compare Typed Boolean (Structures/DontCompareTypedBoolean ; Suggestions)
  - Dump/CollectClassChanges (Dump/CollectClassChanges ; Dump)
  - Dump/FossilizedMethods (Dump/FossilizedMethods ; Dump)
  - Large Try Block (Exceptions/LargeTryBlock ; Suggestions)
  - Swapped Arguments (Classes/SwappedArguments)
  - Wrong Type For Native PHP Function (Php/WrongTypeForNativeFunction ; Analyze, CI-checks)
- 2.1.4
  - Array_merge Needs Array Of Arrays (Structures/ArrayMergeArrayArray ; Analyze)
  - Call Order (Dump/CallOrder ; Dump)
  - Could Be Float (Typehints/CouldBeFloat ; Typechecks, Typehints)
  - Could Be Integer (Typehints/CouldBeInt ; Typechecks, Typehints)
  - Could Be Iterable (Typehints/CouldBeIterable ; Typechecks, Typehints)
  - Extended Typehints (Complete/ExtendedTypehints ; Complete)
  - Mismatch Properties Typehints (Classes/MismatchProperties)
  - No Need For Triple Equal (Structures/NoNeedForTriple ; Analyze)
  - Php/UseMatch (Php/UseMatch ; CompatibilityPHP74)
- 2.1.3
  - Cyclic References (Classes/CyclicReferences)

  - Protocol lists (Type/Protocols ; Appinfo)
  - Wrong Argument Type (Functions/WrongArgumentType ; Analyze, Typechecks)
- 2.1.2
  - Collect Class Constant Counts (Dump/CollectClassConstantCounts)
  - Collect Local Variable Counts (Dump/CollectLocalVariableCounts ; Dump)
  - Collect Method Counts (Dump/CollectMethodCounts ; Dump)
  - Collect Property Counts (Dump/CollectPropertyCounts ; Dump)
  - Could Be Array Typehint (Typehints/CouldBeArray ; Typehints)
  - Could Be Boolean (Typehints/CouldBeBoolean ; Typehints)
  - Could Be CIT (Typehints/CouldBeCIT ; Typehints)
  - Could Be Callable (Typehints/CouldBeCallable ; Typechecks, Typehints)
  - Could Be Null (Typehints/CouldBeNull ; Typechecks, Typehints)
  - Could Be Parent (Typehints/CouldBeParent ; Typechecks, Typehints)
  - Could Be Self (Typehints/CouldBeSelf ; Typechecks, Typehints)
  - Could Be String (Typehints/CouldBeString ; Typechecks, Typehints)
  - Could Be Void (Typehints/CouldBeVoid ; Typechecks, Typehints)
  - Could Not Type (Typehints/CouldNotType ; Typehints)
  - Double Object Assignation (Structures/DoubleObjectAssignation ; Analyze, ClassReview)
  - Possible Alias Confusion (Namespaces/AliasConfusion ; Suggestions)
  - Safe Phpvariables (Php/SafePhpvars ; Internal)
  - Static Global Variables Confusion (Structures/SGVariablesConfusion ; Suggestions)
  - Too Long A Block (Structures/LongBlock ; Suggestions)
  - Too Much Indented (Functions/TooMuchIndented ; Suggestions)
  - Using Deprecated Method (Functions/UsingDeprecated ; Analyze)
- 2.1.1
  - Check Crypto Key Length (Security/CryptoKeyLength ; Security)
  - Dynamic Self Calls (Classes/DynamicSelfCalls)
  - Keep Files Access Restricted (Security/KeepFilesRestricted ; Security)
  - OpenSSL Ciphers Used (Type/OpensslCipher ; Inventory)
  - Prefix And Suffixes With Typehint (Functions/PrefixToType ; Semantics)
  - Throw Was An Expression (Php/ThrowWasAnExpression ; CompatibilityPHP72, CompatibilityPHP73, CompatibilityPHP74)
  - Undefined Constant Name (Variables/UndefinedConstantName ; Analyze)
  - Unused Trait In Class (Traits/UnusedClassTrait ; ClassReview)
- 2.1.0
  - Fn Argument Variable Confusion (Functions/FnArgumentVariableConfusion ; Analyze, Semantics)

  - Hidden Nullable (Classes/HiddenNullable)
  - Missing Abstract Method (Classes/MissingAbstractMethod ; Analyze, ClassReview)
  - Signature Trailing Comma (Php/SignatureTrailingComma ; CompatibilityPHP71, CompatibilityPHP72, CompatibilityPHP73, CompatibilityPHP74)
- 2.0.9
  - Dont Collect Void (Functions/DontUseVoid ; Analyze)
  - Php 8.0 Only TypeHints (Php/Php80OnlyTypeHints ; Appinfo, CompatibilityPHP56, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72, CompatibilityPHP73, CompatibilityPHP74)
  - Uninitilized Property (Classes/UninitedProperty)
  - Union Typehint (Php/Php80UnionTypehint ; Appinfo, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72, CompatibilityPHP73, CompatibilityPHP74)
  - Wrong Typed Property Default (Classes/WrongTypedPropertyInit ; Analyze, LintButWontExec, ClassReview, CI-checks)
- 2.0.8
  - New Functions In PHP 8.0 (Php/Php80NewFunctions)
  - Php 8.0 Variable Syntax Tweaks (Php/Php80VariableSyntax ; Appinfo, CompatibilityPHP74)
- 2.0.7
  - Constant Order (Dump/ConstantOrder)
- 2.0.6
  - Fossilized Method (Classes/FossilizedMethod)
  - Links Between Parameter And Argument (Dump/ParameterArgumentsLinks ; Appinfo)
  - Not Equal Is Not !== (Structures/NotEqual ; Analyze, CI-checks)
  - Possible Interfaces (Interfaces/PossibleInterfaces ; Internal)
- 2.0.5
  - Missing Typehint (Functions/MissingTypehint)
  - Semantic Typing (Functions/SemanticTyping ; Semantics)
- 2.0.4
  - Coalesce Equal (Php/CoalesceEqual)
- 2.0.3
  - Collect Class Children Count (Dump/CollectClassChildren)
  - Collect Class Depth (Dump/CollectClassDepth ; Dump)
  - Collect Class Interface Counts (Dump/CollectClassInterfaceCounts ; Dump)
  - Exceeding Typehint (Functions/ExceedingTypehint ; ClassReview)
- 2.0.2
  - Dump/Inclusions (Dump/Inclusions ; Dump)
  - Dump/NewOrder (Dump/NewOrder ; Dump)
  - Insufficient Property Typehint (Classes/InsufficientPropertyTypehint)

  - Nullable Without Check (Functions/NullableWithoutCheck ; ClassReview)
  - Typehint Order (Dump/TypehintOrder ;)
  - Wrong Typehinted Name (Functions/WrongTypehintedName ; Coding Conventions, Semantics)
- 1.9.9
  - Collect Mbstring Encodings (Dump/CollectMbstringEncodings ; Dump)
  - Complete/CreateForeachDefault (Complete/CreateForeachDefault ; Complete)
  - Concrete usage (Vendors/Concrete5 ; Appinfo)
  - Could Type With Array (Functions/CouldTypeWithArray ; Under Work)
  - Could Type With Boolean (Functions/CouldTypeWithBool ; Under Work)
  - Could Type With Int (Functions/CouldTypeWithInt ; Under Work)
  - Could Type With Iterable (Functions/CouldTypeWithIterable ; Under Work)
  - Could Type With String (Functions/CouldTypeWithString ; Under Work)
  - Filter To add_slashes() (Php/FilterToAddSlashes ; CompatibilityPHP74)
  - Immutable Signature (Classes/ImmutableSignature ; Appinfo)
  - Is_A() With String (Php/IsAWithString ; Analyze, Simple, Rector, CI-checks)
  - Mbstring Third Arg (Structures/MbstringThirdArg ; Analyze, CI-checks)
  - Mbstring Unknown Encoding (Structures/MbstringUnknownEncoding ; Analyze, CI-checks)
  - Merge If Then (Structures/MergeIfThen ; Analyze, CI-checks)
  - Shell commands (Type/Shellcommands ; Appinfo)
  - Typehinting Stats (Dump/TypehintingStats ; Dump)
  - Typo 3 usage (Vendors/Typo3 ; Appinfo)
  - Weird Array Index (Arrays/WeirdIndex)
  - Wrong Case Namespaces (Namespaces/WrongCase ; Coding Conventions)
  - Wrong Type With Call (Functions/WrongTypeWithCall ; Analyze, Typechecks, CI-checks)
- 1.9.8
  - Cant Implement Traversable (Interfaces/CantImplementTraversable ; Analyze, LintButWontExec, CI-checks)
  - Parameter Hiding (Functions/ParameterHiding ; Semantics)
  - Propagate Calls (Complete/PropagateCalls)
- 1.9.7
  - Foreach() Favorite (Dump/CollectForeachFavorite ; Dump)
  - Make Functioncall With Reference (Complete/MakeFunctioncallWithReference ; Complete)
  - Too Many Dereferencing (Classes/TooManyDereferencing)
  - Use Url Query Functions (Structures/UseUrlQueryFunctions ; Suggestions)
- 1.9.6
  - Collect Parameter Counts (Dump/CollectParameterCounts ; Dump)

  - Create Magic Method (Complete/CreateMagicMethod ;)
  - Custom/NotInThisList (Custom/NotInThisList ; Under Work)
  - Dump/DereferencingLevels (Dump/DereferencingLevels ; Dump)
  - Duplicate Literal (Type/DuplicateLiteral ; Semantics)
  - Internet Domains (Type/UdpDomains ; Inventory)
  - No Weak SSL Crypto (Security/NoWeakSSLCrypto ; Security)
  - No mb_substr In Loop (Performances/MbStringInLoop ; Performances)
  - Non Nullable Getters (Classes/NonNullableSetters)
  - Use Case Value (Structures/UseCaseValue ; Suggestions)

- 1.9.5
  - Collect Literals (Dump/CollectLiterals ; Dump)
  - Environment Variable Usage (Dump/EnvironnementVariables ; Dump)
  - Interfaces Don't Ensure Properties (Interfaces/NoGaranteeForPropertyConstant ; Analyze, ClassReview)
  - Interfaces Is Not Implemented (Interfaces/IsNotImplemented ; Analyze, LintButWontExec, ClassReview, CI-checks)
  - Magic Properties (Classes/MagicProperties)
  - No Literal For Reference (Functions/NoLiteralForReference ; Analyze, CI-checks)
  - Use array_slice() (Performances/UseArraySlice ; Analyze, CI-checks)
- 1.9.4
  - Coalesce And Concat (Structures/CoalesceAndConcat ; Analyze, CI-checks)
  - Constant Comparison (Structures/AlwaysFalse ; Analyze)
  - Cyclomatic Complexity (Dump/CyclomaticComplexity ; Dump)
  - Nested Ternary Without Parenthesis (Php/NestedTernaryWithoutParenthesis ; Appinfo, CompatibilityPHP74)
  - PHP 74 New Directives (Php/Php74NewDirective ; CompatibilityPHP73)
  - Should Use Explode Args (Structures/ShouldUseExplodeArgs ; Analyze, CI-checks)
  - Spread Operator For Array (Php/SpreadOperatorForArray ; Appinfo)
  - Too Many Array Dimensions (Arrays/TooManyDimensions)
  - Use Arrow Functions (Functions/UseArrowFunctions ; Appinfo)
- 1.9.3
  - Complete/SetClassRemoteDefinitionWithParenthesis (Complete/SetClassRemoteDefinitionWithParenthesis ; Complete)
  - Complete/SetClassRemoteDefinitionWithTypehint (Complete/SetClassRemoteDefinitionWithTypehint ; Complete)
  - Environment Variables (Dump/EnvironmentVariables ;)
  - Indentation Levels (Dump/IndentationLevels ; Dump)
  - Max Level Of Nesting (Structures/MaxLevelOfIdentation ; Analyze)

  - No Spread For Hash (Arrays/NoSpreadForHash)
  - PHP 7.4 Constant Deprecation (Php/Php74Deprecation ; CompatibilityPHP74)
  - PHP 7.4 Removed Directives (Php/Php74RemovedDirective ; CompatibilityPHP74)
  - Set Class Method Remote Definition (Complete/SetClassMethodRemoteDefinition ; Complete)
  - Set Class Property Definition With Typehint (Complete/SetClassPropertyDefinitionWithTypehint ; Complete)
  - Set Class Remote Definition With Global (Complete/SetClassRemoteDefinitionWithGlobal ; Complete)
  - Set Class Remote Definition With Local New (Complete/SetClassRemoteDefinitionWithLocalNew ; Complete)
  - Set Class Remote Definition With Return Typehint (Complete/SetClassRemoteDefinitionWithReturnTypehint ; Complete)
  - Set String Method Definition (Complete/SetStringMethodDefinition ; Complete)
  - SetA rray Class Definition (Complete/SetArrayClassDefinition ; Complete)
  - Use Contravariance (Php/UseContravariance ; Appinfo)
  - Use Covariance (Php/UseCovariance ; Appinfo)
  - openssl_random_pseudo_byte() Second Argument (Structures/OpensslRandomPseudoByteSecondArg ; CompatibilityPHP74)
  - strip_tags Skips Closed Tag (Structures/StripTagsSkipsClosedTag ; Analyze, CI-checks)

- 1.9.2
  - Complete/SetClassRemoteDefinitionWithInjection (Complete/SetClassRemoteDefinitionWithInjection ; Complete)
  - Create Compact Variables (Complete/CreateCompactVariables)
  - Create Default Values (Complete/CreateDefaultValues ; Complete)
  - Create Magic Property (Complete/CreateMagicProperty ; Complete)
  - Follow Closure Definition (Complete/FollowClosureDefinition ; Complete)
  - Implode() Arguments Order (Structures/ImplodeArgsOrder ; Analyze, CI-checks)
  - Make Class Constant Definition (Complete/MakeClassConstantDefinition ; Complete)
  - Make Class Method Definition (Complete/MakeClassMethodDefinition ; Complete)
  - No ENT_IGNORE (Security/NoEntIgnore ; Security)
  - No More Curly Arrays (Php/NoMoreCurlyArrays ; CompatibilityPHP74)
  - Overwritten Constant (Complete/OverwrittenConstants ; Complete)
  - Overwritten Methods (Complete/OverwrittenMethods ; Complete)
  - Overwritten Properties (Complete/OverwrittenProperties ; Complete)
  - PHP 7.4 Reserved Keyword (Php/Php74ReservedKeyword ; CompatibilityPHP74)
  - Propagate Constants (Complete/PropagateConstants ; Complete)
  - Set Class_Alias Definition (Complete/SetClassAliasDefinition ; Complete)
  - Set Clone Link (Complete/SetCloneLink ; Complete)
  - Set Parent Definition (Complete/SetParentDefinition ; Complete)

- Solve Trait Methods (Complete/SolveTraitMethods ; Complete)
- array_merge() And Variadic (Structures/ArrayMergeAndVariadic ; Analyze)

- 1.9.1
  - Complete/PhpNativeReference (Complete/PhpNativeReference)
- 1.9.0
  - Class Without Parent (Classes/NoParent)
  - Numeric Literal Separator (Php/IntegerSeparatorUsage ; Appinfo, CompatibilityPHP73)
  - PHP 7.4 Removed Functions (Php/Php74RemovedFunctions ; CompatibilityPHP74)
  - Reflection Export() Is Deprecated (Php/ReflectionExportIsDeprecated ; CompatibilityPHP74)
  - Scalar Are Not Arrays (Php/ScalarAreNotArrays ; Analyze, CompatibilityPHP74, CI-checks)
  - Serialize Magic Method (Php/SerializeMagic ; Internal)
  - Similar Integers (Type/SimilarIntegers ; Coding Conventions, Semantics)
  - Unbinding Closures (Functions/UnbindingClosures ; CompatibilityPHP74)
  - array_key_exists() Works On Arrays (Php/ArrayKeyExistsWithObjects ; Analyze, CompatibilityPHP74)
- 1.8.9
  - Avoid mb_dectect_encoding() (Php/AvoidMbDectectEncoding ; Analyze)
  - Disconnected Classes (Classes/DisconnectedClasses)
  - Not Or Tilde (Structures/NotOrNot ; Preferences)
  - Overwritten Source And Value (Structures/ForeachSourceValue ; Analyze, OneFile)
  - Useless Type Check (Functions/UselessTypeCheck ; Dead code, OneFile)
  - mb_strrpos() Third Argument (Php/Php74mbstrrpos3rdArg ; CompatibilityPHP74)
- 1.8.8
  - Set Aside Code (Structures/SetAside)
  - Use Array Functions (Structures/UseArrayFunctions ; Suggestions)
- 1.8.7
  - Cant Use Function (Functions/CantUse)
  - Generator Cannot Return (Functions/GeneratorCannotReturn ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)
  - Use DateTimeImmutable Class (Php/UseDateTimeImmutable ; Suggestions)
  - Wrong Returned Type (Functions/WrongReturnedType ; Analyze, ClassReview, CI-checks)
- 1.8.6
  - Dependant Abstract Classes (Classes/DependantAbstractClass ; Analyze, ClassReview)
  - Infinite Recursion (Structures/InfiniteRecursion ; Analyze)
  - Modules/IncomingData (Modules/IncomingData ; Internal)
  - Modules/NativeReplacement (Modules/NativeReplacement ; Internal)
  - Null Or Boolean Arrays (Arrays/NullBoolean)

- 1.8.5
  - Could Use Trait (Traits/CouldUseTrait)
- 1.8.4
  - Always Use Function With array_key_exists() (Performances/Php74ArrayKeyExists ; Performances)
  - Complex Dynamic Names (Variables/ComplexDynamicNames ; Suggestions)
  - Could Be Constant (Constants/CouldBeConstant ; Suggestions)
  - New Constants In PHP 7.4 (Php/Php74NewConstants ; CompatibilityPHP74)
  - Regex On Arrays (Performances/RegexOnArrays ; Performances)
  - Unused Class Constant (Classes/UnusedConstant)
  - curl_version() Has No Argument (Structures/CurlVersionNow ; CompatibilityPHP74)
- 1.8.3
  - Autoappend (Performances/Autoappend ; Performances)
  - Make Magic Concrete (Classes/MakeMagicConcrete)
  - Memoize MagicCall (Performances/MemoizeMagicCall ; Analyze, ClassReview)
  - Substr To Trim (Structures/SubstrToTrim ; Suggestions)
- 1.8.2
  - Identical Methods (Classes/IdenticalMethods)
  - No Append On Source (Structures/NoAppendOnSource ; Analyze)
- 1.8.1
  - No Need For get_class() (Structures/NoNeedGetClass)
- 1.8.0
  - Already Parents Trait (Traits/AlreadyParentsTrait ; Analyze)
  - Casting Ternary (Structures/CastingTernary ; Analyze, OneFile, CI-checks)
  - Concat And Addition (Php/ConcatAndAddition ; Analyze, CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72, CompatibilityPHP73, CompatibilityPHP74, Top10, CompatibilityPHP80, CI-checks)
  - Concat Empty String (Structures/ConcatEmpty ; Analyze, OneFile)
  - Minus One On Error (Security/MinusOneOnError ; Security)
  - New Functions In PHP 7.4 (Php/Php74NewFunctions ; CompatibilityPHP74)
  - Unpacking Inside Arrays (Php/UnpackingInsideArrays ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72, CompatibilityPHP73)
  - Useless Argument (Functions/UselessArgument)
- 1.7.9
  - Avoid option arrays in constructors (Classes/AvoidOptionArrays)
  - Trait Not Found (Traits/TraitNotFound ; Analyze, LintButWontExec)
  - Useless Default Argument (Functions/UselessDefault ; Suggestions)

  - ext/ffi (Extensions/Extffi ; Appinfo, Appcontent)
  - ext/uuid (Extensions/Extuuid ; Appinfo)
  - ext/zend_monitor (Extensions/Extzendmonitor ; Appinfo)
- 1.7.8
  - ext/svm (Extensions/Extsvm)
- 1.7.7
  - Implode One Arg (Php/ImplodeOneArg)
  - Incoming Values (Php/IncomingValues ; Internal)
  - Integer Conversion (Security/IntegerConversion ; Security)
- 1.7.6
  - Caught Variable (Exceptions/CatchE)
  - Multiple Unset() (Structures/MultipleUnset ; Suggestions, php-cs-fixable)
  - PHP Overridden Function (Php/OveriddenFunction ; Appinfo)
  - array_merge With Ellipsis (Structures/ArrayMergeWithEllipsis ;)
- 1.7.2
  - Check On __Call Usage (Classes/CheckOnCallUsage)
  - Unsupported Operand Types (Structures/UnsupportedOperandTypes ;)
- 1.7.0
  - Clone With Non-Object (Classes/CloneWithNonObject)
  - Self-Transforming Variables (Variables/SelfTransform ; Internal)
  - Should Deep Clone (Classes/ShouldDeepClone ; Suggestions)
  - Windows Only Constants (Portability/WindowsOnlyConstants ;)
- 1.6.9
  - Inconsistent Variable Usage (Variables/InconsistentUsage ; Under Work)
  - Typehint Must Be Returned (Functions/TypehintMustBeReturned)
- 1.6.8
  - PHP 8.0 Removed Constants (Php/Php80RemovedConstant)
  - PHP 8.0 Removed Functions (Php/Php80RemovedFunctions ; CompatibilityPHP80)
- 1.6.7
  - An OOP Factory (Patterns/Factory ; Appinfo)
  - Constant Dynamic Creation (Constants/DynamicCreation ; Appinfo)
  - Law of Demeter (Classes/DemeterLaw)
- 1.6.6
  - Bad Typehint Relay (Functions/BadTypehintRelay)
  - Insufficient Typehint (Functions/InsufficientTypehint ; Analyze, Typechecks)
- 1.6.5

  - String Initialization (Arrays/StringInitialization)
  - Variable Is Not A Condition (Structures/NoVariableIsACondition ; Analyze)
  - ext/pcov (Extensions/Extpcov ; Appinfo)
  - ext/weakref (Extensions/Extweakref ; Appinfo)
- 1.6.4
  - Defined Classes (Modules/DefinedClasses)
  - Don’t Be Too Manual (Structures/DontBeTooManual ; Coding Conventions)
  - Use Coalesce Equal (Structures/UseCoalesceEqual ;)
- 1.6.3
  - Assign And Compare (Structures/AssigneAndCompare)
- 1.6.2
  - Typed Property Usage (Php/TypedPropertyUsage)
- 1.6.1
  - Possible Missing Subpattern (Php/MissingSubpattern ; Analyze, Top10, CI-checks)
  - array_key_exists() Speedup (Performances/ArrayKeyExistsSpeedup)
- 1.5.8
  - Multiple Identical Closure (Functions/MultipleIdenticalClosure)
  - Path lists (Type/Path ; Appinfo)
- 1.5.7
  - Method Could Be Static (Classes/CouldBeStatic)
  - Multiple Usage Of Same Trait (Traits/MultipleUsage ; Suggestions)
  - Self Using Trait (Traits/SelfUsingTrait ; Dead code, ClassReview)
  - ext/wasm (Extensions/Extwasm ; Appinfo)
- 1.5.6
  - Isset() On The Whole Array (Performances/IssetWholeArray ; Performances, Suggestions)
  - Useless Alias (Traits/UselessAlias ; Analyze, LintButWontExec, CI-checks)
  - ext/async (Extensions/Extasync)
  - ext/sdl (Extensions/Extsdl ; Appinfo)
- 1.5.5
  - Directly Use File (Structures/DirectlyUseFile ; Suggestions)
  - Safe HTTP Headers (Security/SafeHttpHeaders ; Security)
  - fputcsv() In Loops (Performances/CsvInLoops)
- 1.5.4
  - Avoid Self In Interface (Interfaces/AvoidSelfInInterface ; ClassReview)
  - Should Have Destructor (Classes/ShouldHaveDestructor)
  - Unreachable Class Constant (Classes/UnreachableConstant ; ClassReview)

- 1.5.3
  - Don't Loop On Yield (Structures/DontLoopOnYield)
  - Variable May Be Non-Global (Structures/VariableMayBeNonGlobal ; Internal)
- 1.5.2
  - PHP Exception (Exceptions/IsPhpException)
  - Should Yield With Key (Functions/ShouldYieldWithKey ; Analyze, Top10, CI-checks)
  - ext/decimal (Extensions/Extdecimal ; Appinfo)
  - ext/psr (Extensions/Extpsr ; Appinfo)
- 1.5.1
  - Use Basename Suffix (Structures/BasenameSuffix)
- 1.5.0
  - Could Use Try (Exceptions/CouldUseTry)
  - Pack Format Inventory (Type/Pack ; Inventory, Appinfo)
  - Printf Format Inventory (Type/Printf ; Inventory, Appinfo)
  - idn_to_ascii() New Default (Php/IdnUts46 ; CompatibilityPHP74)
- 1.4.9
  - Don't Read And Write In One Expression (Structures/DontReadAndWriteInOneExpression ; Analyze, CompatibilityPHP73, CompatibilityPHP74)
  - Invalid Pack Format (Structures/InvalidPackFormat ; Analyze, CI-checks)
  - Named Regex (Structures/NamedRegex ; Suggestions)
  - No Reference For Static Property (Php/NoReferenceForStaticProperty ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72)
  - No Return For Generator (Php/NoReturnForGenerator ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)
  - Repeated Interface (Interfaces/RepeatedInterface ; Analyze, LintButWontExec)
  - Wrong Access Style to Property (Classes/UndeclaredStaticProperty)
- 1.4.8
  - Direct Call To __clone() (Php/DirectCallToClone)
  - filter_input() As A Source (Security/FilterInputSource ; Security)
- 1.4.6
  - Only Variable For Reference (Functions/OnlyVariableForReference)
- 1.4.5
  - Add Default Value (Functions/AddDefaultValue)
- 1.4.4
  - ext/seaslog (Extensions/Extseaslog)
- 1.4.3

  - Class Could Be Final (Classes/CouldBeFinal)
  - Closure Could Be A Callback (Functions/Closure2String ; Performances, Suggestions)
  - Inconsistent Elseif (Structures/InconsistentElseif ; Analyze)
  - Use json_decode() Options (Structures/JsonWithOption ; Suggestions)
- 1.4.2
  - Method Collision Traits (Traits/MethodCollisionTraits)
  - Undefined Insteadof (Traits/UndefinedInsteadof ; Analyze, LintButWontExec, CI-checks)
  - Undefined Variable (Variables/UndefinedVariable ; Analyze, CI-checks)
- 1.4.1
  - Must Call Parent Constructor (Php/MustCallParentConstructor)
- 1.4.0
  - PHP 7.3 Removed Functions (Php/Php73RemovedFunctions)
  - Trailing Comma In Calls (Php/TrailingComma ; Appinfo, CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72)
- 1.3.9
  - Assert Function Is Reserved (Php/AssertFunctionIsReserved ; Analyze, CompatibilityPHP73)
  - Avoid Real (Php/AvoidReal ; Suggestions, Top10)
  - Case Insensitive Constants (Constants/CaseInsensitiveConstants ; Appinfo, CompatibilityPHP73)
  - Const Or Define Preference (Constants/ConstDefinePreference ; Preferences)
  - Continue Is For Loop (Structures/ContinueIsForLoop ; Analyze, CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72, CompatibilityPHP73)
  - Could Be Abstract Class (Classes/CouldBeAbstractClass)
- 1.3.8
  - Constant Case Preference (Constants/DefineInsensitivePreference)
  - Detect Current Class (Php/DetectCurrentClass ; Suggestions, CompatibilityPHP74)
  - Use is_countable (Php/CouldUseIsCountable ; Suggestions)
- 1.3.7
  - Handle Arrays With Callback (Arrays/WithCallback)
- 1.3.5
  - Locally Used Property In Trait (Traits/LocallyUsedProperty ; Internal)
  - PHP 7.0 Scalar Typehints (Php/PHP70scalartypehints ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)
  - PHP 7.1 Scalar Typehints (Php/PHP71scalartypehints ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70)
  - PHP 7.2 Scalar Typehints (Php/PHP72scalartypehints ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70, CompatibilityPHP71)

  - Undefined ::class (Classes/UndefinedStaticclass)
  - ext/lzf (Extensions/Extlzf ; Appinfo)
  - ext/msgpack (Extensions/Extmsgpack ; Appinfo)
- 1.3.4
  - Ambiguous Visibilities (Classes/AmbiguousVisibilities)
  - Hash Algorithms Incompatible With PHP 7.1- (Php/HashAlgos71 ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70)
  - Hash Algorithms Incompatible With PHP 7.4- (Php/HashAlgos74 ; CompatibilityPHP74)
  - ext/csprng (Extensions/Extcsprng ; Appinfo)
- 1.3.3
  - Abstract Or Implements (Classes/AbstractOrImplements)
  - Can't Throw Throwable (Exceptions/CantThrow ; Analyze, LintButWontExec)
  - Incompatible Signature Methods (Classes/IncompatibleSignature ; Analyze, LintButWontExec)
  - Incompatible Signature Methods With Covariance (Classes/IncompatibleSignature74 ; Analyze)
  - ext/eio (Extensions/Exteio ; Appinfo)
- 1.3.2
  - > Or < Comparisons (Structures/GtOrLtFavorite ; Preferences)
  - Compared But Not Assigned Strings (Structures/ComparedButNotAssignedStrings ; Under Work)
  - Could Be Static Closure (Functions/CouldBeStaticClosure)
  - Dont Mix ++ (Structures/DontMixPlusPlus ; Analyze)
  - Strict Or Relaxed Comparison (Structures/ComparisonFavorite ; Preferences)
  - move_uploaded_file Instead Of copy (Security/MoveUploadedFile ; Security)
- 1.3.0
  - Check JSON (Structures/CheckJson ; Analyze, CI-checks)
  - Const Visibility Usage (Classes/ConstVisibilityUsage)
  - Should Use Operator (Structures/ShouldUseOperator ; Suggestions)
  - Single Use Variables (Variables/UniqueUsage ; Under Work)
- 1.2.9
  - Compact Inexistant Variable (Php/CompactInexistant ; CompatibilityPHP73, Suggestions)
  - Configure Extract (Security/ConfigureExtract ; Security)
  - Flexible Heredoc (Php/FlexibleHeredoc ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72)
  - Method Signature Must Be Compatible (Classes/MethodSignatureMustBeCompatible)
  - Mismatch Type And Default (Functions/MismatchTypeAndDefault ; Analyze, LintButWontExec, Typechecks)
  - Use The Blind Var (Performances/UseBlindVar ; Performances)
- 1.2.8

  - Cache Variable Outside Loop (Performances/CacheVariableOutsideLoop ; Performances)
  - Cant Instantiate Class (Classes/CantInstantiateClass)
  - Do In Base (Performances/DoInBase ; Performances)
  - Php/FailingAnalysis (Php/FailingAnalysis ; Internal)
  - Typehinted References (Functions/TypehintedReferences ; Analyze, CI-checks)
  - Weak Typing (Classes/WeakType ; Analyze)
  - strpos() Too Much (Performances/StrposTooMuch ; Analyze, CI-checks)
- 1.2.7
  - ext/cmark (Extensions/Extcmark)
- 1.2.6
  - Callback Needs Return (Functions/CallbackNeedsReturn)
  - Could Use array_unique (Structures/CouldUseArrayUnique ; Suggestions)
  - Missing Parenthesis (Structures/MissingParenthesis ; Analyze, Simple, Level 5, CI-checks)
  - One If Is Sufficient (Structures/OneIfIsSufficient ; Suggestions)
- 1.2.5
  - Wrong Range Check (Structures/WrongRange ; Analyze)
  - ext/zookeeper (Extensions/Extzookeeper)
- 1.2.4
  - Processing Collector (Performances/RegexOnCollector)
- 1.2.3
  - Don’t Unset Properties (Classes/DontUnsetProperties)
  - Redefined Private Property (Classes/RedefinedPrivateProperty ; Analyze)
  - Strtr Arguments (Php/StrtrArguments ; Analyze, CI-checks)
- 1.2.2
  - Drop Substr Last Arg (Structures/SubstrLastArg)
- 1.2.1
  - Possible Increment (Structures/PossibleIncrement ; Suggestions)
  - Properties Declaration Consistence (Classes/PPPDeclarationStyle)
- 1.1.10
  - Too Many Native Calls (Php/TooManyNativeCalls)
- 1.1.9
  - Should Preprocess Chr() (Php/ShouldPreprocess ; Suggestions)
  - Too Many Parameters (Functions/TooManyParameters)
- 1.1.8
  - Mass Creation Of Arrays (Arrays/MassCreation)
  - ext/db2 (Extensions/Extdb2 ; Appinfo)

- 1.1.7
    - Could Use array_fill_keys (Structures/CouldUseArrayFillKeys ; Suggestions)
    - Dynamic Library Loading (Security/DynamicDl ; Security)
    - PHP 7.3 Last Empty Argument (Php/PHP73LastEmptyArgument ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72)
    - Property Could Be Local (Classes/PropertyCouldBeLocal)
    - Use Count Recursive (Structures/UseCountRecursive ; Suggestions)
    - ext/leveldb (Extensions/Extleveldb ; Appinfo)
    - ext/opencensus (Extensions/Extopencensus ; Appinfo)
    - ext/uopz (Extensions/Extuopz ; Appinfo)
    - ext/varnish (Extensions/Extvarnish ; Appinfo)
    - ext/xxtea (Extensions/Extxxtea ; Appinfo)
- 1.1.6
    - Could Use Compact (Structures/CouldUseCompact ; Suggestions)
    - Foreach On Object (Php/ForeachObject)
    - List With Reference (Php/ListWithReference ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72)
    - Test Then Cast (Structures/TestThenCast ; Analyze)
- 1.1.5
    - Possible Infinite Loop (Structures/PossibleInfiniteLoop ; Analyze)
    - Should Use Math (Structures/ShouldUseMath ; Suggestions)
    - ext/hrtime (Extensions/Exthrtime)
- 1.1.4
    - Double array_flip() (Performances/DoubleArrayFlip ; Performances)
    - Fallback Function (Functions/FallbackFunction ; Appinfo)
    - Find Key Directly (Structures/GoToKeyDirectly ; Under Work)
    - Reuse Variable (Structures/ReuseVariable ; Suggestions)
    - Useless Catch (Exceptions/UselessCatch)
- 1.1.3
    - Useless Referenced Argument (Functions/UselessReferenceArgument)
- 1.1.2
    - Local Globals (Variables/LocalGlobals ;)
    - Missing Include (Files/MissingInclude)
- 1.1.1
    - Inclusion Wrong Case (Files/InclusionWrongCase)
- 1.0.11

  - No Net For Xml Load (Security/NoNetForXmlLoad ; Security)
  - Unused Inherited Variable In Closure (Functions/UnusedInheritedVariable)
- 1.0.10
  - Sqlite3 Requires Single Quotes (Security/Sqlite3RequiresSingleQuotes)
- 1.0.8
  - Identical Consecutive Expression (Structures/IdenticalConsecutive ; Analyze)
  - Identical On Both Sides (Structures/IdenticalOnBothSides ; Analyze, CI-checks)
  - Mistaken Concatenation (Arrays/MistakenConcatenation)
  - No Reference For Ternary (Php/NoReferenceForTernary ; Analyze, CI-checks)
- 1.0.7
  - Not A Scalar Type (Php/NotScalarType)
  - Should Use array_filter() (Php/ShouldUseArrayFilter ; Suggestions)
- 1.0.6
  - Never Used Parameter (Functions/NeverUsedParameter ; Analyze, Suggestions)
  - Use Named Boolean In Argument Definition (Functions/AvoidBooleanArgument ; Analyze)
  - ext/igbinary (Extensions/Extigbinary)
- 1.0.5
  - Assigned In One Branch (Structures/AssignedInOneBranch ; Under Work)
  - Environment Variables (Variables/UncommonEnvVar ; Appinfo)
  - Invalid Regex (Structures/InvalidRegex ; Analyze, CI-checks)
  - Parent First (Classes/ParentFirst)
  - Same Variable Foreach (Structures/AutoUnsetForeach ; Analyze, CI-checks)
- 1.0.4
  - Argon2 Usage (Php/Argon2Usage ; Appinfo, Appcontent)
  - Array Index (Type/ArrayIndex ; Inventory, Appinfo)
  - Avoid set_error_handler $context Argument (Php/AvoidSetErrorHandlerContextArg ; CompatibilityPHP72)
  - Can't Count Non-Countable (Structures/CanCountNonCountable ; CompatibilityPHP72)
  - Crypto Usage (Php/CryptoUsage ; Appinfo, Appcontent)
  - Dl() Usage (Php/DlUsage ; Appinfo)
  - Don't Send $this In Constructor (Classes/DontSendThisInConstructor ; Analyze)
  - Hash Will Use Objects (Php/HashUsesObjects ; CompatibilityPHP72)
  - Incoming Variable Index Inventory (Type/GPCIndex ; Inventory, Appinfo, Appcontent)
  - Integer As Property (Classes/IntegerAsProperty ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70, CompatibilityPHP71)
  - Missing New ? (Structures/MissingNew ; Analyze)

  - No get_class() With Null (Structures/NoGetClassNull ; Analyze, CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72)
  - Php 7.2 New Class (Php/Php72NewClasses ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72)
  - Php 7.4 New Class (Php/Php74NewClasses ; CompatibilityPHP74)
  - Slice Arrays First (Arrays/SliceFirst)
  - Unknown Pcre2 Option (Php/UnknownPcre2Option ; Analyze, CompatibilityPHP73)
  - Use List With Foreach (Structures/UseListWithForeach ; Suggestions, Top10)
  - Use PHP7 Encapsed Strings (Performances/PHP7EncapsedStrings ; Performances)
  - ext/vips (Extensions/Extvips ; Appinfo, Appcontent)
- 1.0.3
  - Ambiguous Static (Classes/AmbiguousStatic)
  - Drupal Usage (Vendors/Drupal ; Appinfo)
  - FuelPHP Usage (Vendors/Fuel ; Appinfo, Appcontent)
  - Phalcon Usage (Vendors/Phalcon ; Appinfo)
- 1.0.1
  - Could Be Else (Structures/CouldBeElse ; Analyze)
  - Next Month Trap (Structures/NextMonthTrap ; Analyze, Top10, CI-checks)
  - Printf Number Of Arguments (Structures/PrintfArguments ; Analyze, CI-checks)
  - Simple Switch (Performances/SimpleSwitch)
  - Substring First (Performances/SubstrFirst ; Performances, Suggestions, Top10)
- 0.12.17
  - Is A PHP Magic Property (Classes/IsaMagicProperty)
- 0.12.16
  - Cookies Variables (Php/CookiesVariables)
  - Date Formats (Php/DateFormats ; Inventory)
  - Incoming Variables (Php/IncomingVariables ; Inventory)
  - Session Variables (Php/SessionVariables ; Inventory)
  - Too Complex Expression (Structures/ComplexExpression ; Appinfo)
  - Unconditional Break In Loop (Structures/UnconditionLoopBreak ; Analyze, Level 3, CI-checks)
- 0.12.15
  - Always Anchor Regex (Security/AnchorRegex)
  - Is Actually Zero (Structures/IsZero ; Analyze, Level 2, CI-checks)
  - Multiple Type Variable (Structures/MultipleTypeVariable ; Analyze, Level 4)
  - Session Lazy Write (Security/SessionLazyWrite ; Security)
- 0.12.14

  - Regex Inventory (Type/Regex ; Inventory, Appinfo, Appcontent)
  - Switch Fallthrough (Structures/Fallthrough ; Inventory, Security, Stats)
  - Upload Filename Injection (Security/UploadFilenameInjection)
- 0.12.12
  - Use pathinfo() Arguments (Php/UsePathinfoArgs ; Performances)
  - ext/parle (Extensions/Extparle)
- 0.12.11
  - Could Be Protected Class Constant (Classes/CouldBeProtectedConstant ; ClassReview)
  - Could Be Protected Method (Classes/CouldBeProtectedMethod ; ClassReview)
  - Method Could Be Private Method (Classes/CouldBePrivateMethod)
  - Method Used Below (Classes/MethodUsedBelow ;)
  - Pathinfo() Returns May Vary (Php/PathinfoReturns ; Analyze, Level 4)
- 0.12.10
  - Constant Used Below (Classes/ConstantUsedBelow)
  - Could Be Private Class Constant (Classes/CouldBePrivateConstante ; ClassReview)
- 0.12.9
  - Shell Favorite (Php/ShellFavorite)
- 0.12.8
  - ext/fam (Extensions/Extfam)
  - ext/rdkafka (Extensions/Extrdkafka ; Appinfo)
- 0.12.7
  - Should Use Foreach (Structures/ShouldUseForeach)
- 0.12.5
  - Logical To in_array (Performances/LogicalToInArray)
  - No Substr Minus One (Php/NoSubstrMinusOne ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70)
- 0.12.4
  - Assign With And (Php/AssignAnd ; Analyze, CI-checks)
  - Avoid Concat In Loop (Performances/NoConcatInLoop ; Performances, Top10)
  - Child Class Removes Typehint (Classes/ChildRemoveTypehint)
  - Isset Multiple Arguments (Php/IssetMultipleArgs ; Suggestions, php-cs-fixable)
  - Logical Operators Favorite (Php/LetterCharsLogicalFavorite ; Preferences, Top10)
  - No Magic With Array (Classes/NoMagicWithArray ; Analyze, Level 4, LintButWontExec, CI-checks)
  - Optional Parameter (Functions/OptionalParameter ; DefensiveProgrammingTM)
  - PHP 7.2 Object Keyword (Php/Php72ObjectKeyword ; CompatibilityPHP72)
  - ext/xattr (Extensions/Extxattr ; Appinfo)

- 0.12.3
  - Group Use Trailing Comma (Php/GroupUseTrailingComma ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70, CompatibilityPHP71)
  - Mismatched Default Arguments (Functions/MismatchedDefaultArguments ; Analyze, Typechecks)
  - Mismatched Typehint (Functions/MismatchedTypehint ; Analyze, Typechecks)
  - Scalar Or Object Property (Classes/ScalarOrObjectProperty)
- 0.12.2
  - Mkdir Default (Security/MkdirDefault ; Security)
  - ext/lapack (Extensions/Extlapack)
  - strict_types Preference (Php/DeclareStrict ; Appinfo, Preferences)
- 0.12.1
  - Const Or Define (Structures/ConstDefineFavorite ; Appinfo)
  - Declare strict_types Usage (Php/DeclareStrictType ; Appinfo, Preferences)
  - Encoding Usage (Php/DeclareEncoding)
  - Mismatched Ternary Alternatives (Structures/MismatchedTernary ; Analyze, Suggestions, Level 4)
  - No Return Or Throw In Finally (Structures/NoReturnInFinally ; Security)
  - Ticks Usage (Php/DeclareTicks ; Appinfo, Preferences)
- 0.12.0
  - Avoid Optional Properties (Classes/AvoidOptionalProperties)
  - Heredoc Delimiter (Structures/HeredocDelimiterFavorite ; Coding Conventions)
  - Multiple Functions Declarations (Functions/MultipleDeclarations ; Appinfo)
  - Non Breakable Space In Names (Structures/NonBreakableSpaceInNames ; Appinfo, Appcontent)
  - ext/swoole (Extensions/Extswoole ; Appinfo)
- 0.11.8
  - Cant Inherit Abstract Method (Classes/CantInheritAbstractMethod)
  - Codeigniter usage (Vendors/Codeigniter ; Appinfo)
  - Ez cms usage (Vendors/Ez ; Appinfo)
  - Joomla usage (Vendors/Joomla ; Appinfo, Appcontent)
  - Laravel usage (Vendors/Laravel ; Appinfo, Appcontent)
  - Symfony usage (Vendors/Symfony ; Appinfo)
  - Use session_start() Options (Php/UseSessionStartOptions ; Suggestions)
  - Wordpress usage (Vendors/Wordpress ; Appinfo)
  - Yii usage (Vendors/Yii ; Appinfo, Appcontent)
- 0.11.7
  - Forgotten Interface (Interfaces/CouldUseInterface ; Analyze)
  - Order Of Declaration (Classes/OrderOfDeclaration)

- 0.11.6
  - Concatenation Interpolation Consistence (Structures/ConcatenationInterpolationFavorite ; Preferences)
  - Could Make A Function (Functions/CouldCentralize ; Analyze, Suggestions)
  - Courier Anti-Pattern (Patterns/CourrierAntiPattern ; Appinfo, Appcontent, Dismell)
  - DI Cyclic Dependencies (Classes/TypehintCyclicDependencies ; Dismell)
  - Dependency Injection (Patterns/DependencyInjection ; Appinfo)
  - PSR-13 Usage (Psr/Psr13Usage ; Appinfo)
  - PSR-16 Usage (Psr/Psr16Usage ; Appinfo)
  - PSR-3 Usage (Psr/Psr3Usage ; Appinfo)
  - PSR-6 Usage (Psr/Psr6Usage ; Appinfo)
  - PSR-7 Usage (Psr/Psr7Usage ; Appinfo)
  - Too Many Injections (Classes/TooManyInjections)
  - ext/gender (Extensions/Extgender ; Appinfo)
  - ext/judy (Extensions/Extjudy ; Appinfo)
- 0.11.5
  - Could Typehint (Functions/CouldTypehint ; Under Work)
  - Implemented Methods Are Public (Classes/ImplementedMethodsArePublic)
  - Mixed Concat And Interpolation (Structures/MixedConcatInterpolation ; Analyze, Coding Conventions)
  - No Reference On Left Side (Structures/NoReferenceOnLeft ; Analyze, CI-checks)
  - PSR-11 Usage (Psr/Psr11Usage ; Appinfo)
  - ext/stats (Extensions/Extstats ; Appinfo)
- 0.11.4
  - No Class As Typehint (Functions/NoClassAsTypehint)
  - Use Browscap (Php/UseBrowscap ; Appinfo)
  - Use Debug (Structures/UseDebug ; Appinfo)
- 0.11.3
  - No Return Used (Functions/NoReturnUsed ; Analyze, Suggestions, Level 4)
  - Only Variable Passed By Reference (Functions/OnlyVariablePassedByReference ; Analyze)
  - Try With Multiple Catch (Php/TryMultipleCatch ; Appinfo)
  - ext/grpc (Extensions/Extgrpc)
  - ext/sphinx (Extensions/Extsphinx ; Appinfo)
- 0.11.2
  - Alternative Syntax Consistence (Structures/AlternativeConsistenceByFile ; Analyze)
  - Randomly Sorted Arrays (Arrays/RandomlySortedLiterals)
- 0.11.1
  - Difference Consistence (Structures/DifferencePreference)

  - No Empty Regex (Structures/NoEmptyRegex ; Analyze, CI-checks)
- 0.11.0
  - Could Use str_repeat() (Structures/CouldUseStrrepeat ; Analyze, Level 1, Top10, CI-checks)
  - Crc32() Might Be Negative (Php/Crc32MightBeNegative ; Analyze, PHP recommendations)
  - Empty Final Element (Arrays/EmptyFinal)
  - Strings With Strange Space (Type/StringWithStrangeSpace ; Analyze, CI-checks)
  - Suspicious Comparison (Structures/SuspiciousComparison ; Analyze, Level 3)
- 0.10.9
  - Displays Text (Php/Prints ; Internal)
  - Method Is Overwritten (Classes/MethodIsOverwritten)
  - No Class In Global (Php/NoClassInGlobal ; Analyze, CI-checks)
  - Repeated Regex (Structures/RepeatedRegex ; Analyze, Level 1, CI-checks)
- 0.10.7
  - Group Use Declaration (Php/GroupUseDeclaration)
  - Missing Cases In Switch (Structures/MissingCases ; Analyze)
  - New Constants In PHP 7.2 (Php/Php72NewConstants ; CompatibilityPHP72)
  - New Functions In PHP 7.2 (Php/Php72NewFunctions ; CompatibilityPHP72)
  - New Functions In PHP 7.3 (Php/Php73NewFunctions ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72, CompatibilityPHP73)
- 0.10.6
  - Check All Types (Structures/CheckAllTypes ; Analyze)
  - Do Not Cast To Int (Php/NoCastToInt ;)
  - Manipulates INF (Php/IsINF)
  - Manipulates NaN (Php/IsNAN ; Appinfo)
  - Set Cookie Safe Arguments (Security/SetCookieArgs ; Security)
  - Should Use SetCookie() (Php/UseSetCookie ; Analyze)
  - Use Cookies (Php/UseCookies ; Appinfo, Appcontent)
- 0.10.5
  - Could Be Typehinted Callable (Functions/CouldBeCallable ; Under Work)
  - Encoded Simple Letters (Security/EncodedLetters ; Security)
  - Regex Delimiter (Structures/RegexDelimiter ; Preferences)
  - Strange Name For Constants (Constants/StrangeName ; Analyze)
  - Strange Name For Variables (Variables/StrangeName ; Analyze)
  - Too Many Finds (Classes/TooManyFinds)
- 0.10.4
  - No Need For Else (Structures/NoNeedForElse ; Analyze)

  - Should Use session_regenerateid() (Security/ShouldUseSessionRegenerateId ; Security)
  - ext/ds (Extensions/Extds)
- 0.10.3
  - Multiple Alias Definitions Per File (Namespaces/MultipleAliasDefinitionPerFile ; Analyze, CI-checks)
  - Property Used In One Method Only (Classes/PropertyUsedInOneMethodOnly ; Analyze)
  - Used Once Property (Classes/UsedOnceProperty ; Analyze)
  - __DIR__ Then Slash (Structures/DirThenSlash ; Analyze, Level 3, CI-checks)
  - self, parent, static Outside Class (Classes/NoPSSOutsideClass)
- 0.10.2
  - Class Function Confusion (Php/ClassFunctionConfusion ; Semantics)
  - Forgotten Thrown (Exceptions/ForgottenThrown)
  - Should Use array_column() (Php/ShouldUseArrayColumn ; Performances, Suggestions, Level 4)
  - ext/libsodium (Extensions/Extlibsodium ; Appinfo, Appcontent)
- 0.10.1
  - All strings (Type/CharString ; Inventory)
  - SQL queries (Type/Sql ; Inventory, Appinfo)
  - Strange Names For Methods (Classes/StrangeName)
- 0.10.0
  - Error_Log() Usage (Php/ErrorLogUsage ; Appinfo)
  - No Boolean As Default (Functions/NoBooleanAsDefault ; Analyze)
  - Raised Access Level (Classes/RaisedAccessLevel)
- 0.9.9
  - PHP 7.2 Deprecations (Php/Php72Deprecation)
  - PHP 7.2 Removed Functions (Php/Php72RemovedFunctions ; CompatibilityPHP72)
- 0.9.8
  - Assigned Twice (Variables/AssignedTwiceOrMore ; Analyze)
  - New Line Style (Structures/NewLineStyle ; Preferences)
  - New On Functioncall Or Identifier (Classes/NewOnFunctioncallOrIdentifier)
- 0.9.7
  - Avoid Large Array Assignation (Structures/NoAssignationInFunction ; Performances)
  - Could Be Protected Property (Classes/CouldBeProtectedProperty)
  - Long Arguments (Structures/LongArguments ; Analyze)
- 0.9.6
  - Avoid glob() Usage (Performances/NoGlob ; Performances)
  - Fetch One Row Format (Performances/FetchOneRowFormat)
- 0.9.5

  - One Expression Brackets Consistency (Structures/OneExpressionBracketsConsistency ; Preferences)
  - Should Use Function (Php/ShouldUseFunction ; Performances)
  - ext/mongodb (Extensions/Extmongodb)
  - ext/zbarcode (Extensions/Extzbarcode ; Appinfo)
- 0.9.4
  - Class Should Be Final By Ocramius (Classes/FinalByOcramius)
  - String (Extensions/Extstring ; Appinfo, Appcontent)
  - ext/mhash (Extensions/Extmhash ; Appinfo, CompatibilityPHP54, Appcontent)
- 0.9.3
  - Close Tags Consistency (Php/CloseTagsConsistency)
  - Unset() Or (unset) (Php/UnsetOrCast ; Preferences)
- 0.9.2
  - $GLOBALS Or global (Php/GlobalsVsGlobal ; Preferences)
  - Illegal Name For Method (Classes/WrongName)
  - Too Many Local Variables (Functions/TooManyLocalVariables ; Analyze)
  - Use Composer Lock (Composer/UseComposerLock ; Appinfo)
  - ext/ncurses (Extensions/Extncurses ; Appinfo)
  - ext/newt (Extensions/Extnewt ; Appinfo)
  - ext/nsapi (Extensions/Extnsapi ; Appinfo)
- 0.9.1
  - Avoid Using stdClass (Php/UseStdclass ; Analyze, OneFile, Simple, Level 4)
  - Avoid array_push() (Performances/AvoidArrayPush)
  - Invalid Octal In String (Type/OctalInString ; Inventory, CompatibilityPHP71)
- 0.9.0
  - Getting Last Element (Arrays/GettingLastElement)
  - Rethrown Exceptions (Exceptions/Rethrown ; Dead code)
- 0.8.9
  - Array() / [ ] Consistence (Arrays/ArrayBracketConsistence)
  - Bail Out Early (Structures/BailOutEarly ; Analyze, OneFile, Simple, Level 4)
  - Die Exit Consistence (Structures/DieExitConsistance ; Preferences)
  - Dont Change The Blind Var (Structures/DontChangeBlindKey ; Analyze)
  - More Than One Level Of Indentation (Structures/OneLevelOfIndentation ; Calisthenics)
  - One Dot Or Object Operator Per Line (Structures/OneDotOrObjectOperatorPerLine ; Calisthenics)
  - PHP 7.1 Microseconds (Php/Php71microseconds ; CompatibilityPHP71)
  - Unitialized Properties (Classes/UnitializedProperties ; OneFile, Simple, Suggestions, Level 4, Top10)
  - Useless Check (Structures/UselessCheck ; Analyze, OneFile, Simple, Level 1, CI-checks)

- 0.8.7
  - Don’t Echo Error (Security/DontEchoError ; Analyze, Security, Simple, Level 1, CI-checks)
  - No isset() With empty() (Structures/NoIssetWithEmpty ; Analyze, PHP recommendations, OneFile, RadwellCodes, Simple, Level 4, CI-checks)
  - Use Class Operator (Classes/UseClassOperator)
  - Useless Casting (Structures/UselessCasting ; Analyze, PHP recommendations, OneFile, RadwellCodes, Simple, Level 4, CI-checks)
  - ext/rar (Extensions/Extrar ; Appinfo)
  - time() Vs strtotime() (Performances/timeVsstrtotime ; Performances, OneFile, RadwellCodes)
- 0.8.6
  - Drop Else After Return (Structures/DropElseAfterReturn)
  - Modernize Empty With Expression (Structures/ModernEmpty ; Analyze, OneFile, Simple)
  - Use Positive Condition (Structures/UsePositiveCondition ; Analyze, OneFile, Simple)
- 0.8.5
  - Should Make Ternary (Structures/ShouldMakeTernary ; Analyze, OneFile, Simple, CI-checks)
  - Unused Returned Value (Functions/UnusedReturnedValue)
- 0.8.4
  - $HTTP_RAW_POST_DATA Usage (Php/RawPostDataUsage ; Appinfo, CompatibilityPHP56)
  - $this Belongs To Classes Or Traits (Classes/ThisIsForClasses ; Analyze, Simple)
  - $this Is Not An Array (Classes/ThisIsNotAnArray ; Analyze)
  - $this Is Not For Static Methods (Classes/ThisIsNotForStatic ; Analyze)
  - ** For Exponent (Php/NewExponent ; Suggestions, php-cs-fixable)
  - ::class (Php/StaticclassUsage ; CompatibilityPHP54, CompatibilityPHP53)
  - <?= Usage (Php/EchoTagUsage ; Appinfo, Simple)
  - @ Operator (Structures/Noscream ; Analyze, Appinfo, Performances, ClearPHP, CI-checks)
  - Abstract Class Usage (Classes/Abstractclass ; Appinfo, Appcontent)
  - Abstract Methods Usage (Classes/Abstractmethods ; Appinfo, Appcontent)
  - Abstract Static Methods (Classes/AbstractStatic ; Analyze, Simple)
  - Access Protected Structures (Classes/AccessProtected ; Analyze, Simple)
  - Accessing Private (Classes/AccessPrivate ; Analyze, Simple)
  - Adding Zero (Structures/AddZero ; Analyze, OneFile, ClearPHP, Simple, Level 1, CI-checks)
  - Aliases (Namespaces/Alias ; Appinfo)
  - Aliases Usage (Functions/AliasesUsage ; Analyze, OneFile, ClearPHP, Simple, Level 1, CI-checks)
  - All Uppercase Variables (Variables/VariableUppercase ; Coding Conventions)
  - Already Parents Interface (Interfaces/AlreadyParentsInterface ; Analyze, Suggestions, Level 3)
  - Altering Foreach Without Reference (Structures/AlteringForeachWithoutReference ; Analyze, ClearPHP, Simple, Level 1, CI-checks)

- Alternative Syntax (Php/AlternativeSyntax ; Appinfo)
- Always Positive Comparison (Structures/NeverNegative ; Analyze, Simple, CI-checks)
- Ambiguous Array Index (Arrays/AmbiguousKeys)
- Anonymous Classes (Classes/Anonymous ; Appinfo, CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)
- Argument Should Be Typehinted (Functions/ShouldBeTypehinted ; Typechecks)
- Array Index (Arrays/Arrayindex ; Appinfo)
- Assertions (Php/AssertionUsage ; Appinfo)
- Assign Default To Properties (Classes/MakeDefault ; Analyze, ClearPHP, Simple, Level 2)
- Autoloading (Php/AutoloadUsage ; Appinfo)
- Avoid Parenthesis (Structures/PrintWithoutParenthesis ; Analyze, Simple, CI-checks)
- Avoid Substr() One (Structures/NoSubstrOne ; Analyze, Performances, CompatibilityPHP71, Simple, Suggestions, Level 2, Top10, CI-checks)
- Avoid Those Hash Functions (Security/AvoidThoseCrypto ; Security)
- Avoid array_unique() (Structures/NoArrayUnique ; Performances)
- Avoid get_class() (Structures/UseInstanceof ; Analyze, Simple, CI-checks)
- Avoid sleep()/usleep() (Security/NoSleep ; Security)
- Bad Constants Names (Constants/BadConstantnames ; Analyze, PHP recommendations)
- Binary Glossary (Type/Binary ; Inventory, Appinfo, CompatibilityPHP53)
- Blind Variables (Variables/Blind ;)
- Bracketless Blocks (Structures/Bracketless ; Coding Conventions)
- Break Outside Loop (Structures/BreakOutsideLoop ; Analyze, CompatibilityPHP70)
- Break With 0 (Structures/Break0 ; CompatibilityPHP53, OneFile)
- Break With Non Integer (Structures/BreakNonInteger ; CompatibilityPHP54, OneFile)
- Buried Assignation (Structures/BuriedAssignation ; Analyze)
- Calltime Pass By Reference (Structures/CalltimePassByReference ; CompatibilityPHP54)
- Can't Disable Class (Security/CantDisableClass ; Appinfo)
- Can't Disable Function (Security/CantDisableFunction ; Appinfo, Appcontent)
- Can't Extend Final (Classes/CantExtendFinal ; Analyze, Dead code, Simple)
- Cant Use Return Value In Write Context (Php/CantUseReturnValueInWriteContext ; CompatibilityPHP54, CompatibilityPHP53)
- Cast To Boolean (Structures/CastToBoolean ; Analyze, OneFile, Simple, Level 1)
- Cast Usage (Php/CastingUsage ; Appinfo)
- Catch Overwrite Variable (Structures/CatchShadowsVariable ; Analyze, ClearPHP, Simple)
- Caught Exceptions (Exceptions/CaughtExceptions ;)
- Caught Expressions (Php/TryCatchUsage ; Appinfo)

- Class Const With Array (Php/ClassConstWithArray ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP53)
- Class Has Fluent Interface (Classes/HasFluentInterface ;)
- Class Usage (Classes/ClassUsage ;)
- Class, Interface Or Trait With Identical Names (Classes/CitSameName ; Analyze)
- Classes Mutually Extending Each Other (Classes/MutualExtension ; LintButWontExec, ClassReview)
- Classes Names (Classes/Classnames ; Appinfo)
- Clone Usage (Classes/CloningUsage ; Appinfo)
- Close Tags (Php/CloseTags ; Coding Conventions)
- Closure May Use $this (Php/ClosureThisSupport ; CompatibilityPHP53)
- Closures Glossary (Functions/Closures ; Appinfo)
- Coalesce (Php/Coalesce ; Appinfo, Appcontent)
- Common Alternatives (Structures/CommonAlternatives ; Analyze, Simple)
- Compare Hash (Security/CompareHash ; Security, ClearPHP)
- Compared Comparison (Structures/ComparedComparison ; Analyze)
- Composer Namespace (Composer/IsComposerNsname ; Appinfo, Internal)
- Composer Usage (Composer/UseComposer ; Appinfo)
- Composer’s autoload (Composer/Autoload ; Appinfo)
- Concrete Visibility (Interfaces/ConcreteVisibility ; Analyze, Simple, LintButWontExec)
- Conditional Structures (Structures/ConditionalStructures ;)
- Conditioned Constants (Constants/ConditionedConstants ; Appinfo, Internal)
- Conditioned Function (Functions/ConditionedFunctions ; Appinfo, Internal)
- Confusing Names (Variables/CloseNaming ; Under Work)
- Const With Array (Php/ConstWithArray ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP53)
- Constant Class (Classes/ConstantClass ; Analyze, Simple, CI-checks)
- Constant Comparison (Structures/ConstantComparisonConsistance ; Coding Conventions, Preferences)
- Constant Conditions (Structures/ConstantConditions ;)
- Constant Definition (Classes/ConstantDefinition ; Appinfo, Stats)
- Constant Scalar Expression (Php/ConstantScalarExpression ;)
- Constant Scalar Expressions (Structures/ConstantScalarExpression ; Appinfo, CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP53)
- Constants (Constants/Constantnames ; Inventory, Stats)
- Constants Created Outside Its Namespace (Constants/CreatedOutsideItsNamespace ; Analyze)
- Constants Usage (Constants/ConstantUsage ; Appinfo)
- Constants With Strange Names (Constants/ConstantStrangeNames ; Analyze, Simple, CI-checks)
- Constructors (Classes/Constructor ; Internal)

- Continents (Type/Continents ;)
- Could Be Class Constant (Classes/CouldBeClassConstant ; ClassReview)
- Could Be Static (Structures/CouldBeStatic ; Analyze, OneFile, ClassReview)
- Could Use Alias (Namespaces/CouldUseAlias ; OneFile, Suggestions)
- Could Use Short Assignation (Structures/CouldUseShortAssignation ; Analyze, Performances, OneFile, Simple, CI-checks)
- Could Use __DIR__ (Structures/CouldUseDir ; Analyze, Simple, Suggestions, Level 3, php-cs-fixable, CI-checks)
- Could Use self (Classes/ShouldUseSelf ; Analyze, Simple, Suggestions, Level 3, ClassReview)
- Custom Class Usage (Classes/AvoidUsing ; Custom)
- Custom Constant Usage (Constants/CustomConstantUsage ;)
- Dangling Array References (Structures/DanglingArrayReferences ; Analyze, PHP recommendations, ClearPHP, Simple, Level 1, Top10, CI-checks)
- Deep Definitions (Functions/DeepDefinitions ; Analyze, Appinfo, Simple)
- Define With Array (Php/DefineWithArray ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)
- Defined Class Constants (Classes/DefinedConstants ; Internal)
- Defined Exceptions (Exceptions/DefinedExceptions ; Appinfo)
- Defined Parent MP (Classes/DefinedParentMP ; Internal)
- Defined Properties (Classes/DefinedProperty ; Internal)
- Defined static:: Or self:: (Classes/DefinedStaticMP ; Internal)
- Definitions Only (Files/DefinitionsOnly ; Internal)
- Dependant Trait (Traits/DependantTrait ; Analyze, Level 3)
- Deprecated Functions (Php/Deprecated ; Analyze, CI-checks)
- Dereferencing String And Arrays (Structures/DereferencingAS ; Appinfo, CompatibilityPHP54, CompatibilityPHP53)
- Direct Injection (Security/DirectInjection ; Security)
- Directives Usage (Php/DirectivesUsage ; Appinfo)
- Don’t Change Incomings (Structures/NoChangeIncomingVariables ; Analyze)
- Double Assignation (Structures/DoubleAssignation ; Analyze)
- Double Instructions (Structures/DoubleInstruction ; Analyze, Simple)
- Duplicate Calls (Structures/DuplicateCalls ;)
- Dynamic Calls (Structures/DynamicCalls ; Appinfo, Internal, Stats)
- Dynamic Class Constant (Classes/DynamicConstantCall ; Appinfo)
- Dynamic Classes (Classes/DynamicClass ; Appinfo)
- Dynamic Code (Structures/DynamicCode ; Appinfo)
- Dynamic Function Call (Functions/Dynamiccall ; Appinfo, Internal, Stats)
- Dynamic Methodcall (Classes/DynamicMethodCall ; Appinfo)

- Dynamic New (Classes/DynamicNew ; Appinfo)
- Dynamic Property (Classes/DynamicPropertyCall ; Appinfo)
- Dynamically Called Classes (Classes/VariableClasses ; Appinfo, Stats)
- Echo Or Print (Structures/EchoPrintConsistance ; Coding Conventions, Preferences)
- Echo With Concat (Structures/EchoWithConcat ; Analyze, Performances, Simple, Suggestions)
- Ellipsis Usage (Php/EllipsisUsage ; Appinfo, CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP53)
- Else If Versus Elseif (Structures/ElseIfElseif ; Analyze, Simple, php-cs-fixable, Rector, CI-checks)
- Else Usage (Structures/ElseUsage ; Appinfo, Appcontent, Calisthenics, Stats)
- Email Addresses (Type/Email ; Inventory, Appinfo)
- Empty Blocks (Structures/EmptyBlocks ; Analyze, Simple, CI-checks)
- Empty Classes (Classes/EmptyClass ; Analyze, Simple)
- Empty Function (Functions/EmptyFunction ; Analyze, Simple)
- Empty Instructions (Structures/EmptyLines ; Analyze, Dead code, Simple)
- Empty Interfaces (Interfaces/EmptyInterface ; Analyze, Simple)
- Empty List (Php/EmptyList ; Analyze, CompatibilityPHP70)
- Empty Namespace (Namespaces/EmptyNamespace ; Analyze, Dead code, OneFile, Simple, CI-checks)
- Empty Slots In Arrays (Arrays/EmptySlots ; Coding Conventions)
- Empty Traits (Traits/EmptyTrait ; Analyze, Simple)
- Empty Try Catch (Structures/EmptyTryCatch ; Analyze, Level 3)
- Empty With Expression (Structures/EmptyWithExpression ; OneFile, Suggestions)
- Error Messages (Structures/ErrorMessages ; Appinfo)
- Eval() Usage (Structures/EvalUsage ; Analyze, Appinfo, Security, Performances, OneFile, ClearPHP, Simple)
- Exception Order (Exceptions/AlreadyCaught ; Dead code)
- Exit() Usage (Structures/ExitUsage ; Analyze, Appinfo, OneFile, ClearPHP, CI-checks)
- Exit-like Methods (Functions/KillsApp ; Internal)
- Exponent Usage (Php/ExponentUsage ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP53)
- External Config Files (Files/Services ; Internal)
- Failed Substr Comparison (Structures/FailingSubstrComparison ; Analyze, Simple, Level 3, Top10, CI-checks)
- File Is Component (Files/IsComponent ; Internal)
- File Uploads (Structures/FileUploadUsage ; Appinfo)
- File Usage (Structures/FileUsage ; Appinfo)
- Final Class Usage (Classes/Finalclass ; LintButWontExec, ClassReview)
- Final Methods Usage (Classes/Finalmethod ; LintButWontExec, ClassReview)

- Fopen Binary Mode (Portability/FopenMode ; Portability)
- For Using Functioncall (Structures/ForWithFunctioncall ; Performances, ClearPHP, Simple, Level 1, Top10)
- Foreach Don't Change Pointer (Php/ForeachDontChangePointer ; CompatibilityPHP70)
- Foreach Needs Reference Array (Structures/ForeachNeedReferencedSource ; Under Work)
- Foreach Reference Is Not Modified (Structures/ForeachReferenceIsNotModified ; Analyze, Simple, CI-checks)
- Foreach With list() (Structures/ForeachWithList ; CompatibilityPHP54, CompatibilityPHP53)
- Forgotten Visibility (Classes/NonPpp ; Analyze, ClearPHP, Simple, Level 1, CI-checks)
- Forgotten Whitespace (Structures/ForgottenWhiteSpace ; Analyze, CI-checks)
- Fully Qualified Constants (Namespaces/ConstantFullyQualified ; Analyze)
- Function Called With Other Case Than Defined (Functions/FunctionCalledWithOtherCase ;)
- Function Subscripting (Structures/FunctionSubscripting ; Appinfo, CompatibilityPHP53)
- Function Subscripting, Old Style (Structures/FunctionPreSubscripting ; Suggestions)
- Functioncall Is Global (Functions/IsGlobal ; Under Work)
- Functions Glossary (Functions/Functionnames ; Appinfo)
- Functions In Loop Calls (Functions/LoopCalling ; Under Work)
- Functions Removed In PHP 5.4 (Php/Php54RemovedFunctions ; CompatibilityPHP54)
- Functions Removed In PHP 5.5 (Php/Php55RemovedFunctions ; CompatibilityPHP55)
- Functions Using Reference (Functions/FunctionsUsingReference ; Appinfo, Appcontent)
- GPRC Aliases (Security/GPRAliases ; Internal)
- Global Code Only (Files/GlobalCodeOnly ; Internal)
- Global Import (Namespaces/GlobalImport ; Internal)
- Global In Global (Structures/GlobalInGlobal ; Appinfo)
- Global Inside Loop (Structures/GlobalOutsideLoop ; Performances)
- Global Usage (Structures/GlobalUsage ; Analyze, Appinfo, ClearPHP)
- Globals (Variables/Globals ; Internal)
- Goto Names (Php/Gotonames ; Appinfo, ClearPHP)
- HTTP Status Code (Type/HttpStatus ; Inventory)
- Hardcoded Passwords (Functions/HardcodedPasswords ; Analyze, Security, OneFile, Simple, Level 3)
- Has Magic Property (Classes/HasMagicProperty ; Internal)
- Has Variable Arguments (Functions/VariableArguments ; Appinfo, Internal)
- Hash Algorithms (Php/HashAlgos ; Analyze, Level 4)
- Hash Algorithms Incompatible With PHP 5.3 (Php/HashAlgos53 ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72)
- Hash Algorithms Incompatible With PHP 5.4/5.5 (Php/HashAlgos54 ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72)

- Heredoc Delimiter Glossary (Type/Heredoc ; Appinfo)
- Hexadecimal Glossary (Type/Hexadecimal ; Inventory, Appinfo)
- Hexadecimal In String (Type/HexadecimalString ; Inventory, CompatibilityPHP70, CompatibilityPHP71)
- Hidden Use Expression (Namespaces/HiddenUse ; Analyze, OneFile, Simple, CI-checks)
- Htmlentities Calls (Structures/Htmlentitiescall ; Analyze, Simple, CI-checks)
- Http Headers (Type/HttpHeader ; Inventory)
- Identical Conditions (Structures/IdenticalConditions ; Analyze, Simple, CI-checks)
- If With Same Conditions (Structures/IfWithSameConditions ; Analyze, Simple, CI-checks)
- Iffectations (Structures/Iffectation ; Analyze)
- Implement Is For Interface (Classes/ImplementIsForInterface ; Analyze, Simple)
- Implicit Global (Structures/ImplicitGlobal ;)
- Implied If (Structures/ImpliedIf ; Analyze, ClearPHP, Simple, CI-checks)
- Inclusions (Structures/IncludeUsage ; Appinfo)
- Incompilable Files (Php/Incompilable ; Analyze, Appinfo, ClearPHP, Simple)
- Inconsistent Concatenation (Structures/InconsistentConcatenation ; Internal)
- Indices Are Int Or String (Structures/IndicesAreIntOrString ; Analyze, OneFile, Simple, CI-checks)
- Indirect Injection (Security/IndirectInjection ; Security)
- Instantiating Abstract Class (Classes/InstantiatingAbstractClass ; Analyze, Simple)
- Interface Arguments (Variables/InterfaceArguments ;)
- Interface Methods (Interfaces/InterfaceMethod ;)
- Interfaces Glossary (Interfaces/Interfacenames ; Appinfo)
- Interfaces Usage (Interfaces/InterfaceUsage ;)
- Internally Used Properties (Classes/PropertyUsedInternally ;)
- Internet Ports (Type/Ports ; Inventory)
- Interpolation (Type/StringInterpolation ; Coding Conventions)
- Invalid Constant Name (Constants/InvalidName ; Analyze, Simple)
- Is An Extension Class (Classes/IsExtClass ;)
- Is An Extension Constant (Constants/IsExtConstant ; Internal, First)
- Is An Extension Function (Functions/IsExtFunction ; Internal, First)
- Is An Extension Interface (Interfaces/IsExtInterface ; Internal, First)
- Is CLI Script (Files/IsCliScript ; Appinfo, Internal)
- Is Composer Class (Composer/IsComposerClass ; Internal)
- Is Composer Interface (Composer/IsComposerInterface ; Internal)
- Is Extension Trait (Traits/IsExtTrait ; Internal, First)
- Is Generator (Functions/IsGenerator ; Appinfo, Internal)
- Is Global Constant (Constants/IsGlobalConstant ; Internal)

- Is Interface Method (Classes/IsInterfaceMethod ; Internal)
- Is Library (Project/IsLibrary ;)
- Is Not Class Family (Classes/IsNotFamily ; Internal)
- Is PHP Constant (Constants/IsPhpConstant ; Internal)
- Is Upper Family (Classes/IsUpperFamily ; Internal)
- Joining file() (Performances/JoinFile ; Performances)
- Labels (Php/Labelnames ; Appinfo)
- Linux Only Files (Portability/LinuxOnlyFiles ; Portability)
- List Short Syntax (Php/ListShortSyntax ; Appinfo, CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, Internal, CompatibilityPHP53, CompatibilityPHP70)
- List With Appends (Php/ListWithAppends ; CompatibilityPHP70)
- List With Keys (Php/ListWithKeys ; Appinfo, CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, Appcontent, CompatibilityPHP53, CompatibilityPHP70)
- Locally Unused Property (Classes/LocallyUnusedProperty ; Dead code, Simple)
- Locally Used Property (Classes/LocallyUsedProperty ; Internal)
- Logical Mistakes (Structures/LogicalMistakes ; Analyze, Simple, Level 1, CI-checks)
- Logical Should Use Symbolic Operators (Php/LogicalInLetters ; Analyze, OneFile, ClearPHP, Simple, Suggestions, Level 2, Top10, php-cs-fixable, CI-checks)
- Lone Blocks (Structures/LoneBlock ; Analyze, Simple, Level 4, CI-checks)
- Lost References (Variables/LostReferences ; Analyze, Simple)
- Magic Constant Usage (Constants/MagicConstantUsage ; Appinfo)
- Magic Methods (Classes/MagicMethod ; Appinfo)
- Magic Visibility (Classes/toStringPss ; CompatibilityPHP70, Simple)
- Mail Usage (Structures/MailUsage ; Appinfo)
- Make Global A Property (Classes/MakeGlobalAProperty ; Analyze, Simple)
- Make One Call With Array (Performances/MakeOneCall ; Performances)
- Malformed Octal (Type/MalformedOctal ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)
- Mark Callable (Functions/MarkCallable ; Appinfo, Internal, First)
- Md5 Strings (Type/Md5String ; Inventory, Appinfo)
- Method Has Fluent Interface (Functions/HasFluentInterface ;)
- Method Has No Fluent Interface (Functions/HasNotFluentInterface ;)
- Methodcall On New (Php/MethodCallOnNew ; CompatibilityPHP53)
- Methods Without Return (Functions/WithoutReturn ; Analyze)
- Mime Types (Type/MimeType ; Inventory)
- Mixed Keys Arrays (Arrays/MixedKeys ; CompatibilityPHP54, CompatibilityPHP53)
- Multidimensional Arrays (Arrays/Multidimensional ; Appinfo)

- Multiple Alias Definitions (Namespaces/MultipleAliasDefinitions ; Analyze, Simple, CI-checks)
- Multiple Catch (Structures/MultipleCatch ; Appinfo, Internal)
- Multiple Class Declarations (Classes/MultipleDeclarations ; Analyze, Simple, CI-checks)
- Multiple Classes In One File (Classes/MultipleClassesInFile ; Appinfo, Coding Conventions)
- Multiple Constant Definition (Constants/MultipleConstantDefinition ; Analyze, Simple, CI-checks)
- Multiple Definition Of The Same Argument (Functions/MultipleSameArguments ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, OneFile, ClearPHP, Simple)
- Multiple Exceptions Catch() (Exceptions/MultipleCatch ; Appinfo, CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70)
- Multiple Identical Trait Or Interface (Classes/MultipleTraitOrInterface ; Analyze, OneFile, Simple, CI-checks)
- Multiple Index Definition (Arrays/MultipleIdenticalKeys ; Analyze, OneFile, Simple, CI-checks)
- Multiple Returns (Functions/MultipleReturn ;)
- Multiples Identical Case (Structures/MultipleDefinedCase ; Analyze, OneFile, ClearPHP, Simple, Level 1, CI-checks)
- Multiply By One (Structures/MultiplyByOne ; Analyze, OneFile, ClearPHP, Simple, Level 1, CI-checks)
- Must Return Methods (Functions/MustReturn ; Analyze, Simple, Level 2, LintButWontExec, CI-checks)
- Namespaces (Namespaces/NamespaceUsage ; Appinfo)
- Namespaces Glossary (Namespaces/Namespacesnames ; Appinfo)
- Negative Power (Structures/NegativePow ; Analyze, OneFile, Simple, Level 3, CI-checks)
- Nested Ifthen (Structures/NestedIfthen ; Analyze, RadwellCodes)
- Nested Loops (Structures/NestedLoops ; Appinfo)
- Nested Ternary (Structures/NestedTernary ; Analyze, ClearPHP, Simple, Level 1, CI-checks)
- Never Used Properties (Classes/PropertyNeverUsed ; Analyze, Simple)
- New Functions In PHP 5.4 (Php/Php54NewFunctions ; CompatibilityPHP53)
- New Functions In PHP 5.5 (Php/Php55NewFunctions ; CompatibilityPHP54, CompatibilityPHP53)
- New Functions In PHP 5.6 (Php/Php56NewFunctions ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP53)
- New Functions In PHP 7.0 (Php/Php70NewFunctions ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)
- New Functions In PHP 7.1 (Php/Php71NewFunctions ; CompatibilityPHP71)
- No Choice (Structures/NoChoice ; Analyze, Simple, Level 2, Top10, CI-checks)
- No Count With 0 (Performances/NotCountNull ; Performances)
- No Direct Access (Structures/NoDirectAccess ; Appinfo)
- No Direct Call To Magic Method (Classes/DirectCallToMagicMethod ; Analyze, Level 2, CI-checks)
- No Direct Usage (Structures/NoDirectUsage ; Analyze, Simple)
- No Hardcoded Hash (Structures/NoHardcodedHash ; Analyze, Security, Simple)
- No Hardcoded Ip (Structures/NoHardcodedIp ; Analyze, Security, ClearPHP, Simple)

- No Hardcoded Path (Structures/NoHardcodedPath ; Analyze, ClearPHP, Simple)
- No Hardcoded Port (Structures/NoHardcodedPort ; Analyze, Security, ClearPHP, Simple)
- No List With String (Php/NoListWithString ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)
- No Parenthesis For Language Construct (Structures/NoParenthesisForLanguageConstruct ; Analyze, ClearPHP, RadwellCodes, Simple, Suggestions, Level 2, CI-checks)
- No Plus One (Structures/PlusEgalOne ; Coding Conventions, OneFile)
- No Public Access (Classes/NoPublicAccess ; Analyze)
- No Real Comparison (Type/NoRealComparison ; Analyze, Simple, Level 2, Top10, CI-checks)
- No Self Referencing Constant (Classes/NoSelfReferencingConstant ; Analyze, Simple, LintButWontExec, ClassReview)
- No String With Append (Php/NoStringWithAppend ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)
- No array_merge() In Loops (Performances/ArrayMergeInLoops ; Analyze, Performances, ClearPHP, Simple, Level 2, Top10, CI-checks)
- Non Ascii Variables (Variables/VariableNonascii ; Analyze)
- Non Static Methods Called In A Static (Classes/NonStaticMethodsCalledStatic ; Analyze, CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, Simple, CI-checks)
- Non-constant Index In Array (Arrays/NonConstantArray ; Analyze, Simple)
- Non-lowercase Keywords (Php/UpperCaseKeyword ; Coding Conventions, RadwellCodes)
- Normal Methods (Classes/NormalMethods ; Appcontent)
- Not Definitions Only (Files/NotDefinitionsOnly ; Appinfo)
- Not Not (Structures/NotNot ; Analyze, OneFile, Simple, CI-checks)
- Not Same Name As File (Classes/NotSameNameAsFile ;)
- Not Same Name As File (Classes/SameNameAsFile ; Internal)
- Nowdoc Delimiter Glossary (Type/Nowdoc ; Appinfo)
- Null Coalesce (Php/NullCoalesce ;)
- Null On New (Classes/NullOnNew ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, OneFile, Simple)
- Objects Don't Need References (Structures/ObjectReferences ; Analyze, OneFile, ClearPHP, Simple, Level 2, Top10, CI-checks)
- Octal Glossary (Type/Octal ; Appinfo)
- Old Style Constructor (Classes/OldStyleConstructor ; Analyze, Appinfo, OneFile, ClearPHP, Simple, CompatibilityPHP80)
- Old Style __autoload() (Php/oldAutoloadUsage ; Analyze, OneFile, ClearPHP, Simple)
- One Letter Functions (Functions/OneLetterFunctions ; Coding Conventions, Semantics)
- One Object Operator Per Line (Classes/OneObjectOperatorPerLine ; Calisthenics)
- One Variable String (Type/OneVariableStrings ; Analyze, RadwellCodes, Simple, CI-checks)
- Only Static Methods (Classes/OnlyStaticMethods ; Internal)

- Only Variable Returned By Reference (Structures/OnlyVariableReturnedByReference ; Analyze, Simple)
- Or Die (Structures/OrDie ; Analyze, OneFile, ClearPHP, Simple, CI-checks)
- Overwriting Variable (Variables/Overwriting ;)
- Overwritten Class Const (Classes/OverwrittenConst ; Appinfo)
- Overwritten Exceptions (Exceptions/OverwriteException ; Analyze, Simple, Suggestions, Level 4, CI-checks)
- Overwritten Literals (Variables/OverwrittenLiterals ; Analyze)
- PHP 7.0 New Classes (Php/Php70NewClasses ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)
- PHP 7.0 New Interfaces (Php/Php70NewInterfaces ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)
- PHP 7.0 Removed Directives (Php/Php70RemovedDirective ; CompatibilityPHP70, CompatibilityPHP71)
- PHP 7.0 Removed Functions (Php/Php70RemovedFunctions ; CompatibilityPHP70, Compatibility-PHP71)
- PHP 7.1 Removed Directives (Php/Php71RemovedDirective ; CompatibilityPHP71)
- PHP Arrays Index (Arrays/Phparrayindex ; Appinfo)
- PHP Bugfixes (Php/MiddleVersion ; Appinfo, Appcontent)
- PHP Constant Usage (Constants/PhpConstantUsage ; Appinfo)
- PHP Handlers Usage (Php/SetHandlers ;)
- PHP Interfaces (Interfaces/Php ;)
- PHP Keywords As Names (Php/ReservedNames ; Analyze, Simple)
- PHP Sapi (Type/Sapi ; Internal)
- PHP Variables (Variables/VariablePhp ;)
- PHP5 Indirect Variable Expression (Variables/Php5IndirectExpression ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)
- PHP7 Dirname (Structures/PHP7Dirname ; CompatibilityPHP54, CompatibilityPHP55, Compatibility-PHP56, CompatibilityPHP53, Suggestions, php-cs-fixable)
- Parent, Static Or Self Outside Class (Classes/PssWithoutClass ; Analyze, Simple)
- Parenthesis As Parameter (Php/ParenthesisAsParameter ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)
- Pear Usage (Php/PearUsage ; Appinfo, Appcontent)
- Perl Regex (Type/Pcre ; Inventory)
- Php 7 Indirect Expression (Variables/Php7IndirectExpression ; CompatibilityPHP54, Compatibility-PHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70)
- Php 7.1 New Class (Php/Php71NewClasses ; CompatibilityPHP54, CompatibilityPHP55, Compatibility-PHP56, CompatibilityPHP53, CompatibilityPHP70)
- Php7 Relaxed Keyword (Php/Php7RelaxedKeyword ; Appinfo, CompatibilityPHP54, Compatibility-PHP55, CompatibilityPHP56, CompatibilityPHP53)
- Phpinfo (Structures/PhpinfoUsage ; Security, OneFile, Simple)

- Pre-increment (Performances/PrePostIncrement ; Analyze, Performances, Simple, Level 4, CI-checks)
- Preprocess Arrays (Arrays/ShouldPreprocess ; Suggestions)
- Preprocessable (Structures/ShouldPreprocess ; Analyze, Rector)
- Print And Die (Structures/PrintAndDie ; Analyze, Simple, CI-checks)
- Property Could Be Private Property (Classes/CouldBePrivate ; ClassReview)
- Property Names (Classes/PropertyDefinition ; Internal)
- Property Used Above (Classes/PropertyUsedAbove ; Internal)
- Property Used Below (Classes/PropertyUsedBelow ; Internal)
- Property Variable Confusion (Structures/PropertyVariableConfusion ; Semantics)
- Queries In Loops (Structures/QueriesInLoop ; Analyze, OneFile, Simple, Level 1, Top10)
- Random Without Try (Structures/RandomWithoutTry ; Security)
- Real Functions (Functions/RealFunctions ; Appcontent, Stats)
- Real Variables (Variables/RealVariables ; Appcontent, Stats)
- Recursive Functions (Functions/Recursive ; Appinfo)
- Redeclared PHP Functions (Functions/RedeclaredPhpFunction ; Analyze, Appinfo, Simple, CI-checks)
- Redefined Class Constants (Classes/RedefinedConstants ; Analyze, Simple, CI-checks)
- Redefined Default (Classes/RedefinedDefault ; Analyze, Simple, CI-checks)
- Redefined Methods (Classes/RedefinedMethods ; Appinfo)
- Redefined PHP Traits (Traits/Php ; Appinfo)
- Redefined Property (Classes/RedefinedProperty ; ClassReview)
- References (Variables/References ; Appinfo)
- Register Globals (Security/RegisterGlobals ; Security)
- Relay Function (Functions/RelayFunction ; Analyze)
- Repeated print() (Structures/RepeatedPrint ; Analyze, Simple, Suggestions, Level 3, Top10, CI-checks)
- Reserved Keywords In PHP 7 (Php/ReservedKeywords7 ; CompatibilityPHP70)
- Resources Usage (Structures/ResourcesUsage ; Appinfo)
- Results May Be Missing (Structures/ResultMayBeMissing ; Analyze, Simple, CI-checks)
- Return True False (Structures/ReturnTrueFalse ; Analyze, Simple, Level 1, CI-checks)
- Return Typehint Usage (Php/ReturnTypehintUsage ; Appinfo, Internal)
- Return With Parenthesis (Php/ReturnWithParenthesis ; Coding Conventions, PHP recommendations, Suggestions)
- Return void (Structures/ReturnVoid ;)
- Safe Curl Options (Security/CurlOptions ; Security)
- Same Conditions In Condition (Structures/SameConditions ; Analyze, Simple, CI-checks)
- Scalar Typehint Usage (Php/ScalarTypehintUsage ; Appinfo)
- Sensitive Argument (Security/SensitiveArgument ; Internal)

- Sequences In For (Structures/SequenceInFor ;)
- Setlocale() Uses Constants (Structures/SetlocaleNeedsConstants ; CompatibilityPHP70)
- Several Instructions On The Same Line (Structures/OneLineTwoInstructions ; Analyze)
- Shell Usage (Structures/ShellUsage ; Appinfo)
- Short Open Tags (Php/ShortOpenTagRequired ; Analyze, Simple)
- Short Syntax For Arrays (Arrays/ArrayNSUsage ; Appinfo, CompatibilityPHP53)
- Should Be Single Quote (Type/ShouldBeSingleQuote ; Coding Conventions, ClearPHP)
- Should Chain Exception (Structures/ShouldChainException ; Analyze, Simple, CI-checks)
- Should Make Alias (Namespaces/ShouldMakeAlias ; Analyze, OneFile, Simple, CI-checks)
- Should Typecast (Type/ShouldTypecast ; Analyze, OneFile, Simple, CI-checks)
- Should Use Coalesce (Php/ShouldUseCoalesce ; Analyze, Simple, Suggestions, Level 3, CI-checks)
- Should Use Constants (Functions/ShouldUseConstants ; Analyze, Simple)
- Should Use Local Class (Classes/ShouldUseThis ; Analyze, ClearPHP, Simple)
- Should Use Prepared Statement (Security/ShouldUsePreparedStatement ; Analyze, Security, Simple, CI-checks)
- Silently Cast Integer (Type/SilentlyCastInteger ; Analyze, Simple, CI-checks)
- Simple Global Variable (Php/GlobalWithoutSimpleVariable ; CompatibilityPHP70)
- Simplify Regex (Structures/SimplePreg ; Performances)
- Slow Functions (Performances/SlowFunctions ; Performances, OneFile)
- Special Integers (Type/SpecialIntegers ; Inventory)
- Static Loop (Structures/StaticLoop ; Analyze, Simple, Level 4)
- Static Methods (Classes/StaticMethods ; Appinfo)
- Static Methods Called From Object (Classes/StaticMethodsCalledFromObject ; Analyze, Simple, CI-checks)
- Static Methods Can’t Contain $this (Classes/StaticContainsThis ; Analyze, ClearPHP, Simple, Level 1, CI-checks)
- Static Properties (Classes/StaticProperties ; Appinfo)
- Static Variables (Variables/StaticVariables ; Appinfo)
- Strict Comparison With Booleans (Structures/BooleanStrictComparison ; Analyze, Simple, Suggestions, Level 2, CI-checks)
- String May Hold A Variable (Type/StringHoldAVariable ; Analyze, Simple)
- String glossary (Type/String ;)
- Strpos()-like Comparison (Structures/StrposCompare ; Analyze, PHP recommendations, ClearPHP, Simple, Level 2, Top10, CI-checks)
- Super Global Usage (Php/SuperGlobalUsage ; Appinfo)
- Super Globals Contagion (Security/SuperGlobalContagion ; Internal)
- Switch To Switch (Structures/SwitchToSwitch ; Analyze, RadwellCodes, Simple)

- Switch With Too Many Default (Structures/SwitchWithMultipleDefault ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, ClearPHP, Simple)
- Switch Without Default (Structures/SwitchWithoutDefault ; Analyze, ClearPHP, Simple, CI-checks)
- Ternary In Concat (Structures/TernaryInConcat ; Analyze, Simple, Level 3, CI-checks)
- Test Class (Classes/TestClass ; Appinfo)
- Throw (Php/ThrowUsage ; Appinfo)
- Throw Functioncall (Exceptions/ThrowFunctioncall ; Analyze, Simple, Level 1, CI-checks)
- Throw In Destruct (Classes/ThrowInDestruct ; Analyze, Simple, CI-checks)
- Thrown Exceptions (Exceptions/ThrownExceptions ; Appinfo)
- Throws An Assignement (Structures/ThrowsAndAssign ; Analyze, Simple, CI-checks)
- Timestamp Difference (Structures/TimestampDifference ; Analyze, Simple, Level 3, CI-checks)
- Too Many Children (Classes/TooManyChildren ; Suggestions)
- Trait Methods (Traits/TraitMethod ;)
- Trait Names (Traits/Traitnames ; Appinfo)
- Traits Usage (Traits/TraitUsage ; Appinfo)
- Trigger Errors (Php/TriggerErrorUsage ; Appinfo)
- True False Inconsistant Case (Constants/InconsistantCase ; Preferences)
- Try With Finally (Structures/TryFinally ; Appinfo, Internal)
- Typehints (Functions/Typehints ; Appinfo)
- URL List (Type/Url ; Inventory, Appinfo)
- Uncaught Exceptions (Exceptions/UncaughtExceptions ; Analyze)
- Unchecked Resources (Structures/UncheckedResources ; Analyze, ClearPHP, Simple, Level 2, CI-checks)
- Undefined Caught Exceptions (Exceptions/CaughtButNotThrown ; Dead code)
- Undefined Class Constants (Classes/UndefinedConstants ; Analyze, CI-checks)
- Undefined Classes (Classes/UndefinedClasses ; Analyze)
- Undefined Constants (Constants/UndefinedConstants ; Analyze, CompatibilityPHP72, Simple, CI-checks)
- Undefined Functions (Functions/UndefinedFunctions ; Analyze, CI-checks)
- Undefined Interfaces (Interfaces/UndefinedInterfaces ; Analyze, CI-checks)
- Undefined Parent (Classes/UndefinedParentMP ; Analyze, Simple)
- Undefined Properties (Classes/UndefinedProperty ; Analyze, ClearPHP, Simple, CI-checks)
- Undefined Trait (Traits/UndefinedTrait ; Analyze, LintButWontExec, CI-checks)
- Undefined static:: Or self:: (Classes/UndefinedStaticMP ; Analyze, Simple)
- Unicode Blocks (Type/UnicodeBlock ; Inventory)
- Unicode Escape Partial (Php/UnicodeEscapePartial ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)
- Unicode Escape Syntax (Php/UnicodeEscapeSyntax ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)

- Unknown Directive Name (Php/DirectiveName ; Internal)
- Unkown Regex Options (Structures/UnknownPregOption ; Analyze, Simple)
- Unpreprocessed Values (Structures/Unpreprocessed ; Analyze, OneFile, ClearPHP, Simple)
- Unreachable Code (Structures/UnreachableCode ; Dead code, OneFile, ClearPHP, Simple, Suggestions, Level 3)
- Unresolved Catch (Classes/UnresolvedCatch ; Dead code, ClearPHP)
- Unresolved Classes (Classes/UnresolvedClasses ; Analyze)
- Unresolved Instanceof (Classes/UnresolvedInstanceof ; Analyze, Dead code, ClearPHP, Simple, Top10)
- Unresolved Use (Namespaces/UnresolvedUse ; Analyze, ClearPHP, Simple)
- Unserialize Second Arg (Security/UnserializeSecondArg ; Security)
- Unset Arguments (Functions/UnsetOnArguments ; OneFile)
- Unset In Foreach (Structures/UnsetInForeach ; Analyze, Dead code, OneFile, Simple)
- Unthrown Exception (Exceptions/Unthrown ; Analyze, Dead code, ClearPHP, Simple)
- Unused Arguments (Functions/UnusedArguments ; Analyze, Simple)
- Unused Classes (Classes/UnusedClass ; Analyze, Dead code, Simple)
- Unused Constants (Constants/UnusedConstants ; Dead code, Simple)
- Unused Functions (Functions/UnusedFunctions ; Dead code, Simple)
- Unused Global (Structures/UnusedGlobal ; Analyze, Simple)
- Unused Interfaces (Interfaces/UnusedInterfaces ; Dead code, Simple, Suggestions, Level 2)
- Unused Label (Structures/UnusedLabel ; Dead code, Simple)
- Unused Methods (Classes/UnusedMethods ; Dead code, Simple)
- Unused Private Methods (Classes/UnusedPrivateMethod ; Dead code, OneFile, Simple)
- Unused Private Properties (Classes/UnusedPrivateProperty ; Dead code, OneFile, Simple)
- Unused Protected Methods (Classes/UnusedProtectedMethods ; Dead code)
- Unused Traits (Traits/UnusedTrait ; Simple)
- Unused Use (Namespaces/UnusedUse ; Dead code, ClearPHP, Simple)
- Unusual Case For PHP Functions (Php/UpperCaseFunction ; Coding Conventions)
- Usage Of class_alias() (Classes/ClassAliasUsage ; Appinfo)
- Use === null (Php/IsnullVsEqualNull ; Analyze, OneFile, RadwellCodes, Simple, php-cs-fixable, CI-checks)
- Use Cli (Php/UseCli ; Appinfo)
- Use Const And Functions (Namespaces/UseFunctionsConstants ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP53)
- Use Constant (Structures/UseConstant ; Analyze, PHP recommendations, php-cs-fixable, CI-checks)
- Use Constant As Arguments (Functions/UseConstantAsArguments ; Analyze, Simple, CI-checks)
- Use Instanceof (Classes/UseInstanceof ; Analyze, Simple, CI-checks)

- Use Lower Case For Parent, Static And Self (Php/CaseForPSS ; CompatibilityPHP54, CompatibilityPHP53)
- Use Nullable Type (Php/UseNullableType ; Appinfo, CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53, CompatibilityPHP70)
- Use PHP Object API (Php/UseObjectApi ; Analyze, ClearPHP, Simple, CI-checks)
- Use Pathinfo (Php/UsePathinfo ; Analyze, Simple, Level 3, CI-checks)
- Use System Tmp (Structures/UseSystemTmp ; Analyze, Simple, Level 3, CI-checks)
- Use This (Classes/UseThis ; Internal)
- Use Web (Php/UseWeb ; Appinfo)
- Use With Fully Qualified Name (Namespaces/UseWithFullyQualifiedNS ; Analyze, Coding Conventions, PHP recommendations, Simple)
- Use const (Constants/ConstRecommended ; Analyze, Coding Conventions, Top10, CI-checks)
- Use password_hash() (Php/Password55 ; CompatibilityPHP55)
- Use random_int() (Php/BetterRand ; Analyze, Security, CompatibilityPHP71, Simple, Level 2, CI-checks)
- Used Classes (Classes/UsedClass ; Internal)
- Used Functions (Functions/UsedFunctions ; Internal)
- Used Interfaces (Interfaces/UsedInterfaces ; Internal)
- Used Methods (Classes/UsedMethods ; Internal)
- Used Once Variables (In Scope) (Variables/VariableUsedOnceByContext ; Analyze, OneFile, ClearPHP, Simple, Level 4)
- Used Once Variables (Variables/VariableUsedOnce ; Analyze, OneFile, Simple, Top10)
- Used Private Methods (Classes/UsedPrivateMethod ; Internal)
- Used Protected Method (Classes/UsedProtectedMethod ;)
- Used Static Properties (Classes/UsedPrivateProperty ; Internal)
- Used Trait (Traits/UsedTrait ; Internal)
- Used Use (Namespaces/UsedUse ;)
- Useless Abstract Class (Classes/UselessAbstract ; Analyze, Simple)
- Useless Brackets (Structures/UselessBrackets ; Analyze, RadwellCodes, Simple, CI-checks)
- Useless Constructor (Classes/UselessConstructor ; Analyze, Simple, Level 3)
- Useless Final (Classes/UselessFinal ; Analyze, OneFile, ClearPHP, Simple, CI-checks)
- Useless Global (Structures/UselessGlobal ; Analyze, OneFile, Simple, Level 2)
- Useless Instructions (Structures/UselessInstruction ; Analyze, OneFile, ClearPHP, Simple, Level 1, CI-checks)
- Useless Interfaces (Interfaces/UselessInterfaces ; Analyze, ClearPHP, Simple, ClassReview, Typechecks)
- Useless Parenthesis (Structures/UselessParenthesis ; Analyze, Simple, CI-checks)
- Useless Return (Functions/UselessReturn ; Analyze, OneFile, Simple, Level 4)
- Useless Switch (Structures/UselessSwitch ; Analyze, Simple)
- Useless Unset (Structures/UselessUnset ; Analyze, OneFile, ClearPHP, Simple, Level 2, CI-checks)

- Uses Default Values (Functions/UsesDefaultArguments ; Analyze, Simple, CI-checks)
- Uses Environment (Php/UsesEnv ; Appinfo, Appcontent)
- Using $this Outside A Class (Classes/UsingThisOutsideAClass ; Analyze, CompatibilityPHP71, Simple, LintButWontExec)
- Using Short Tags (Structures/ShortTags ; Appinfo)
- Usort Sorting In PHP 7.0 (Php/UsortSorting ; CompatibilityPHP70)
- Var Keyword (Classes/OldStyleVar ; Analyze, OneFile, ClearPHP, Simple, Level 1)
- Variable Constants (Constants/VariableConstant ; Appinfo, Stats)
- Variables Variables (Variables/VariableVariables ; Appinfo, Stats)
- Variables With Long Names (Variables/VariableLong ; Appinfo)
- Variables With One Letter Names (Variables/VariableOneLetter ; Semantics)
- While(List() = Each()) (Structures/WhileListEach ; Analyze, Performances, OneFile, Simple, Suggestions, Level 2, CI-checks)
- Written Only Variables (Variables/WrittenOnlyVariable ; Analyze, OneFile, Simple)
- Wrong Class Name Case (Classes/WrongCase ; Coding Conventions, RadwellCodes, Simple)
- Wrong Function Name Case (Functions/WrongCase ; Coding Conventions)
- Wrong Number Of Arguments (Functions/WrongNumberOfArguments ; Analyze, OneFile, Simple, CI-checks)
- Wrong Number Of Arguments In Methods (Functions/WrongNumberOfArgumentsMethods ; Under Work)
- Wrong Optional Parameter (Functions/WrongOptionalParameter ; Analyze, Simple, Level 1, CI-checks)
- Wrong Parameter Type (Php/InternalParameterType ; Analyze, OneFile, Simple, CI-checks)
- Wrong fopen() Mode (Php/FopenMode ; Analyze, CI-checks)
- Yield From Usage (Php/YieldFromUsage ; Appinfo, Appcontent)
- Yield Usage (Php/YieldUsage ; Appinfo, Appcontent)
- Yoda Comparison (Structures/YodaComparison ; Coding Conventions)
- __debugInfo() Usage (Php/debugInfoUsage ; CompatibilityPHP54, CompatibilityPHP55, Compatibility-PHP53)
- __halt_compiler (Php/Haltcompiler ; Appinfo)
- __toString() Throws Exception (Structures/toStringThrowsException ; Analyze, OneFile, Simple)
- crypt() Without Salt (Structures/CryptWithoutSalt ; CompatibilityPHP54)
- error_reporting() With Integers (Structures/ErrorReportingWithInteger ; Analyze, Simple, CI-checks)
- eval() Without Try (Structures/EvalWithoutTry ; Analyze, Security, Simple, Level 3, CI-checks)
- ext/0mq (Extensions/Extzmq ; Appinfo)
- ext/amqp (Extensions/Extamqp ; Appinfo)
- ext/apache (Extensions/Extapache ; Appinfo)
- ext/apc (Extensions/Extapc ; Appinfo, CompatibilityPHP55)
- ext/apcu (Extensions/Extapcu ; Appinfo)

- ext/array (Extensions/Extarray ; Appinfo)
- ext/bcmath (Extensions/Extbcmath ; Appinfo)
- ext/bzip2 (Extensions/Extbzip2 ; Appinfo)
- ext/cairo (Extensions/Extcairo ; Appinfo)
- ext/calendar (Extensions/Extcalendar ; Appinfo)
- ext/com (Extensions/Extcom ; Appinfo)
- ext/crypto (Extensions/Extcrypto ; Appinfo)
- ext/ctype (Extensions/Extctype ; Appinfo)
- ext/curl (Extensions/Extcurl ; Appinfo)
- ext/cyrus (Extensions/Extcyrus ; Appinfo)
- ext/date (Extensions/Extdate ; Appinfo)
- ext/dba (Extensions/Extdba ; Appinfo, CompatibilityPHP53)
- ext/dio (Extensions/Extdio ; Appinfo)
- ext/dom (Extensions/Extdom ; Appinfo)
- ext/eaccelerator (Extensions/Exteaccelerator ; Appinfo)
- ext/enchant (Extensions/Extenchant ; Appinfo)
- ext/ereg (Extensions/Extereg ; Appinfo, CompatibilityPHP70)
- ext/ev (Extensions/Extev ; Appinfo)
- ext/event (Extensions/Extevent ; Appinfo)
- ext/exif (Extensions/Extexif ; Appinfo)
- ext/expect (Extensions/Extexpect ; Appinfo)
- ext/fann (Extensions/Extfann ; Appinfo)
- ext/fdf (Extensions/Extfdf ; Appinfo, CompatibilityPHP53)
- ext/ffmpeg (Extensions/Extffmpeg ; Appinfo)
- ext/file (Extensions/Extfile ; Appinfo)
- ext/fileinfo (Extensions/Extfileinfo ; Appinfo)
- ext/filter (Extensions/Extfilter ; Appinfo)
- ext/fpm (Extensions/Extfpm ; Appinfo)
- ext/ftp (Extensions/Extftp ; Appinfo)
- ext/gd (Extensions/Extgd ; Appinfo)
- ext/gearman (Extensions/Extgearman ; Appinfo)
- ext/geoip (Extensions/Extgeoip ; Appinfo)
- ext/gettext (Extensions/Extgettext ; Appinfo)
- ext/gmagick (Extensions/Extgmagick ; Appinfo)
- ext/gmp (Extensions/Extgmp ; Appinfo)
- ext/gnupgp (Extensions/Extgnupg ; Appinfo)

- ext/hash (Extensions/Exthash ; Appinfo)
- ext/ibase (Extensions/Extibase ; Appinfo)
- ext/iconv (Extensions/Exticonv ; Appinfo)
- ext/iis (Extensions/Extiis ; Appinfo, Portability)
- ext/imagick (Extensions/Extimagick ; Appinfo)
- ext/imap (Extensions/Extimap ; Appinfo)
- ext/info (Extensions/Extinfo ; Appinfo)
- ext/inotify (Extensions/Extinotify ; Appinfo)
- ext/intl (Extensions/Extintl ; Appinfo)
- ext/json (Extensions/Extjson ; Appinfo)
- ext/kdm5 (Extensions/Extkdm5 ; Appinfo)
- ext/ldap (Extensions/Extldap ; Appinfo)
- ext/libevent (Extensions/Extlibevent ; Appinfo)
- ext/libxml (Extensions/Extlibxml ; Appinfo)
- ext/lua (Extensions/Extlua ; Appinfo)
- ext/mail (Extensions/Extmail ; Appinfo)
- ext/mailparse (Extensions/Extmailparse ; Appinfo)
- ext/math (Extensions/Extmath ; Appinfo)
- ext/mbstring (Extensions/Extmbstring ; Appinfo)
- ext/mcrypt (Extensions/Extmcrypt ; Appinfo, CompatibilityPHP71)
- ext/memcache (Extensions/Extmemcache ; Appinfo)
- ext/memcached (Extensions/Extmemcached ; Appinfo)
- ext/ming (Extensions/Extming ; Appinfo, CompatibilityPHP53)
- ext/mongo (Extensions/Extmongo ; Appinfo)
- ext/mssql (Extensions/Extmssql ; Appinfo)
- ext/mysql (Extensions/Extmysql ; Appinfo, CompatibilityPHP55)
- ext/mysqli (Extensions/Extmysqli ; Appinfo)
- ext/ob (Extensions/Extob ; Appinfo)
- ext/oci8 (Extensions/Extoci8 ; Appinfo)
- ext/odbc (Extensions/Extodbc ; Appinfo)
- ext/opcache (Extensions/Extopcache ; Appinfo)
- ext/openssl (Extensions/Extopenssl ; Appinfo)
- ext/parsekit (Extensions/Extparsekit ; Appinfo)
- ext/password (Extensions/Extpassword ; Appinfo, Appcontent)
- ext/pcntl (Extensions/Extpcntl ; Appinfo)
- ext/pcre (Extensions/Extpcre ; Appinfo)

  - ext/pdo (Extensions/Extpdo ; Appinfo)
  - ext/pecl_http (Extensions/Exthttp ; Appinfo, Appcontent)
  - ext/pgsql (Extensions/Extpgsql ; Appinfo)
  - ext/phalcon (Extensions/Extphalcon ; Appinfo)
  - ext/phar (Extensions/Extphar ; Appinfo)
  - ext/php-ast (Extensions/Extast ; Appinfo)
  - ext/posix (Extensions/Extposix ; Appinfo)
  - ext/proctitle (Extensions/Extproctitle ; Appinfo)
  - ext/pspell (Extensions/Extpspell ; Appinfo)
  - ext/readline (Extensions/Extreadline ; Appinfo)
  - ext/recode (Extensions/Extrecode ; Appinfo, Portability)
  - ext/redis (Extensions/Extredis ; Appinfo)
  - ext/reflection (Extensions/Extreflection ; Appinfo)
  - ext/runkit (Extensions/Extrunkit ; Appinfo)
  - ext/sem (Extensions/Extsem ; Appinfo)
  - ext/session (Extensions/Extsession ; Appinfo)
  - ext/shmop (Extensions/Extshmop ; Appinfo)
  - ext/simplexml (Extensions/Extsimplexml ; Appinfo)
  - ext/snmp (Extensions/Extsnmp ; Appinfo)
  - ext/soap (Extensions/Extsoap ; Appinfo)
  - ext/sockets (Extensions/Extsockets ; Appinfo)
  - ext/spl (Extensions/Extspl ; Appinfo)
  - ext/sqlite (Extensions/Extsqlite ; Appinfo)
  - ext/sqlite3 (Extensions/Extsqlite3 ; Appinfo)
  - ext/sqlsrv (Extensions/Extsqlsrv ; Appinfo)
  - ext/ssh2 (Extensions/Extssh2 ; Appinfo)
  - ext/standard (Extensions/Extstandard ; Appinfo)
  - ext/suhosin (Extensions/Extsuhosin ; Appinfo)
  - ext/tidy (Extensions/Exttidy ; Appinfo)
  - ext/tokenizer (Extensions/Exttokenizer ; Appinfo)
  - ext/tokyotyrant (Extensions/Exttokyotyrant ; Appinfo)
  - ext/trader (Extensions/Exttrader ; Appinfo)
  - ext/v8js (Extensions/Extv8js ; Appinfo)
  - ext/wddx (Extensions/Extwddx ; Appinfo)
  - ext/wikidiff2 (Extensions/Extwikidiff2 ; Appinfo)
  - ext/wincache (Extensions/Extwincache ; Appinfo, Portability)

  - ext/xcache (Extensions/Extxcache ; Appinfo)
  - ext/xdebug (Extensions/Extxdebug ; Appinfo)
  - ext/xdiff (Extensions/Extxdiff ; Appinfo)
  - ext/xhprof (Extensions/Extxhprof ; Appinfo)
  - ext/xml (Extensions/Extxml ; Appinfo)
  - ext/xmlreader (Extensions/Extxmlreader ; Appinfo)
  - ext/xmlrpc (Extensions/Extxmlrpc ; Appinfo)
  - ext/xmlwriter (Extensions/Extxmlwriter ; Appinfo)
  - ext/xsl (Extensions/Extxsl ; Appinfo)
  - ext/yaml (Extensions/Extyaml ; Appinfo)
  - ext/yis (Extensions/Extyis ; Appinfo)
  - ext/zip (Extensions/Extzip ; Appinfo)
  - ext/zlib (Extensions/Extzlib ; Appinfo)
  - func_get_arg() Modified (Functions/funcGetArgModified ; Analyze, CompatibilityPHP70, Simple)
  - include_once() Usage (Structures/OnceUsage ; Analyze, Appinfo)
  - isset() With Constant (Structures/IssetWithConstant ; CompatibilityPHP54, CompatibilityPHP55, CompatibilityPHP56, CompatibilityPHP53)
  - list() May Omit Variables (Structures/ListOmissions ; Analyze, Simple, Suggestions, Level 3, CI-checks)
  - mcrypt_create_iv() With Default Values (Structures/McryptcreateivWithoutOption ; CompatibilityPHP70)
  - parse_str() Warning (Security/parseUrlWithoutParameters ; Security)
  - preg_match_all() Flag (Php/PregMatchAllFlag ; Simple, Suggestions)
  - preg_replace With Option e (Structures/pregOptionE ; Analyze, Security, CompatibilityPHP70, CompatibilityPHP71, CompatibilityPHP72, Simple, CI-checks)
  - set_exception_handler() Warning (Php/SetExceptionHandlerPHP7 ; CompatibilityPHP70)
  - var_dump()... Usage (Structures/VardumpUsage ; Analyze, Security, ClearPHP, CI-checks)

- 0.8.3
  - Variable Global (Structures/VariableGlobal)

18.8 PHP Error messages

Exakat helps reduce the amount of error and warning that code is producing by reporting pattern that are likely to emit errors.

102 PHP error message detailled :

- :ref:‘ Cannot use parent when current class scope has no parent <class-without-parent>‘
- :ref:‘ Default value for parameters with a int type can only be int or NULL <mismatch-type-and-default>‘
- :ref:‘ array_merge() expects at least 1 parameter, 0 given <array_merge()-and-variadic>‘
- *“continue” targeting switch is equivalent to “break”. Did you mean to use “continue 2”?*

- *A function with return type must return a value (did you mean "return null;" instead of "return;"?)*
- *Access level to Bar::$publicProperty must be public (as in class Foo)*
- *Access level to c::iPrivate() must be public (as in class i)*
- *Access level to x::foo() must be public (as in class i)*
- *Access level to xx::$x must be public (as in class x)*
- *Access to undeclared static property*
- *Accessing static property aa::$a as non static*
- *An alias (%s) was defined for method %s(), but this method does not exist*
- *Argument 1 passed to foo() must be of the type integer, string given*
- *Argument cannot be passed by reference*
- *Argument cannot be passed by reference*
- *Array and string offset access syntax with curly braces is deprecated*
- Call to a member function m() on null
- *Call to undefined function*
- *Call to undefined method theParent::bar()*
- *Can't inherit abstract function A::bar()*
- *Cannot access parent:: when current class scope has no parent*
- *Cannot access parent:: when current class scope has no parent*
- *Cannot access private const*
- *Cannot access static:: when no class scope is active*
- *Cannot override final method Foo::Bar()*
- *Cannot override final method Foo::FooBar()*
- *Cannot pass parameter 1 by reference*
- *Cannot pass parameter 1 by reference*
- *Cannot unpack array with string keys*
- *Cannot use "parent" when no class scope is active*
- *Cannot use "self" when no class scope is active*
- *Cannot use "static" when no class scope is active*
- *Cannot use a scalar value as an array*
- *Cannot use isset() on the result of an expression (you can use "null !== expression" instead)*
- *Cannot use object of type Foo as array*
- Case-insensitive constants are deprecated. The correct casing for this constant is "A"
- *Class 'PARENT' not found*
- *Class 'x' not found*
- *Class BA contains 1 abstract method and must therefore be declared abstract or implement the remaining methods (A::aFoo)*

- *Class b cannot implement previously implemented interface i*
- *Class b cannot implement previously implemented interface i*
- *Class c contains 1 abstract method and must therefore be declared abstract or implement the remaining methods (a::foo)*
- *Class fooThrowable cannot implement interface Throwable, extend Exception or Error instead*
- *Class x contains 2 abstract methods and must therefore be declared abstract or implement the remaining methods (x::m1, x::m2)*
- *Class x must implement interface Traversable as part of either Iterator or IteratorAggregate*
- *Could not check compatibility between xx::bar(B $a) and foo::bar(A $a), because class A is not available*
- *Creating default object from empty value*
- *Declaration of FooParent::Bar() must be compatible with FooChildren::Bar()*
- Declaration of a::foo($a) should be compatible with ab1::foo($a)
- *Declaration of ab::foo($a) must be compatible with a::foo($a = 1)*
- *Declaration of ab::foo($a) must be compatible with a::foo($a = 1)*
- *Declaration of ab::foo($a) should be compatible with a::foo($a = 1)*
- *Declaration of ab::foo($a) should be compatible with a::foo($a = 1)*
- *Defining a custom assert() function is deprecated, as the function has special semantics*
- *Delimiter must not be alphanumeric or backslash*
- *Generators cannot return values using "return"*
- *Generators cannot return values using "return"*
- *Indirect modification of overloaded property c::$b has no effect*
- *Invalid numeric literal*
- *Method name must be a string*
- *Methods with the same name as their class will not be constructors in a future version of PHP; %s has a deprecated constructor*
- *Non-static method A::B() should not be called statically*
- *Old style constructors are DEPRECATED in PHP 7.0, and will be removed in a future version. You should always use __construct() in new code.*
- *Only variable references should be returned by reference*
- *Only variable references should be returned by reference*
- *Only variables can be passed by reference*
- *Only variables should be passed by reference*
- *Return value of foo() must be an instance of Bar, none returned*
- *Return value of foo() must be of the type int, string returned*
- *The (real) cast is deprecated, use (float) instead*
- *The behavior of unparenthesized expressions containing both '.' and '+'/'-' will change in PHP 8: '+'/'-' will take a higher precedence*
- The behavior of unparenthesized expressions containing both '.' and '>>'/'

- *The each() function is deprecated. This message will be suppressed on further calls*
- *The parent constructor was not called: the object is in an invalid state*
- *Too few arguments to function foo(), 1 passed and exactly 2 expected*
- *Trait 'T' not found*
- *Trait 'a' not found*
- *Trait method M has not been applied, because there are collisions with other trait methods on C*
- *Trait method f has not been applied, because there are collisions with other trait methods on x*
- *Trying to access array offset on value of type boolean*
- *Trying to access array offset on value of type float*
- *Trying to access array offset on value of type int*
- *Trying to access array offset on value of type null*
- *Trying to access array offset on value of type null*
- *Uncaught ArgumentCountError: Too few arguments to function, 0 passed*
- *Undefined class constant*
- *Undefined constant 'y'*
- *Undefined function*
- *Undefined variable:*
- *Unknown named parameter $d in*
- *Unparenthesized a ? b : c ? d : e is deprecated. Use either (a ? b : c) ? d : e or a ? b : (c ? d : e)*
- Unsupported operand types
- *Unsupported operand types*
- *Use of undefined constant y - assumed 'y' (this will throw an Error in a future version of PHP)*
- *Using $this when not in object context*
- *__autoload() is deprecated, use spl_autoload_register() instead*
- *__clone method called on non-object*
- *define(): Declaration of case-insensitive constants is deprecated*
- iconv(): Wrong charset, conversion from UTF-8' to ASCII//TRANSLIT' is not allowed
- *pack(): Type t: unknown format code*
- *syntax error, unexpected '-', expecting '='*
- *unpack(): Type t: unknown format code*

18.9 External services

List of external services whose configuration files has been commited in the code.

- Apache - .htaccess, htaccess.txt
- Apple - .DS_Store

- appveyor - appveyor.yml, .appveyor.yml
- ant - build.xml
- apigen - apigen.yml, apigen.neon
- arcunit - .arcunit
- artisan - artisan
- atoum - .bootstrap.atoum.php, .atoum.php, .atoum.bootstrap.php
- arcanist - .arclint, .arcconfig
- bazaar - .bzr
- babeljs - .babel.rc, .babel.js, .babelrc
- behat - behat.yml.dist, behat.yml
- box2 - box.json, box.json.dist
- bower - bower.json, .bowerrc
- circleCI - circle.yml, .circleci
- codacy - .codacy.json
- codeception - codeception.yml, codeception.dist.yml
- codecov - .codecov.yml, codecov.yml
- codeclimate - .codeclimate.yml
- composer - composer.json, composer.lock, vendor
- couscous - couscous.yml
- Code Sniffer - .php_cs, .php_cs.dist, .phpcs.xml, php_cs.dist, phpcs.xml, phpcs.xml.dist
- coveralls - .coveralls.yml
- crowdin - crowdin.yml
- cvs - CVS
- docker - .dockerignore, .docker, docker-compose.yml, Dockerfile
- dotenv - .env.dist, .env, .env.example
- drone - .dockerignore, .docker
- drupalci - drupalci.yml
- drush - drush.services.yml
- editorconfig - .editorconfig
- eslint - .eslintrc, .eslintignore, eslintrc.js, .eslintrc.js, .eslintrc.json
- Exakat - .exakat.yaml, .exakat.yml, .exakat.ini
- flintci - .flintci.yml
- git - .git, .gitignore, .gitattributes, .gitmodules, .mailmap, .githooks
- github - .github
- gitlab - .gitlab-ci.yml
- gulpfile - gulpfile.js

- grumphp - grumphp.yml.dist, grumphp.yml
- gush - .gush.yml
- gruntjs - Gruntfile.js
- humbug - humbug.json.dist, humbug.json
- infection - infection.yml, .infection.yml, infection.json.dist
- insight - .sensiolabs.yml
- jetbrains - .idea
- jshint - .jshintrc, .jshintignore
- mercurial - .hg, .hgtags, .hgignore, .hgeol
- mkdocs - mkdocs.yml
- npm - package.json, .npmignore, .npmrc, package-lock.json
- openshift - .openshift
- phan - .phan
- pharcc - .pharcc.yml
- phalcon - .phalcon
- phpbench - phpbench.json
- phpci - phpci.yml
- Phpdocumentor - .phpdoc.xml, phpdoc.dist.xml
- phpdox - phpdox.xml.dist, phpdox.xml
- phinx - phinx.yml
- phpformatter - .formatter.yml
- phpmetrics - .phpmetrics.yml.dist
- phpsa - .phpsa.yml
- phpspec - phpspec.yml, .phpspec, phpspec.yml.dist
- phpstan - phpstan.neon, .phpstan.neon, phpstan.neon.dist
- phpswitch - .phpswitch.yml
- PHPUnit - phpunit.xml.dist, phpunit.xml
- prettier - .prettierrc, .prettierignore
- psalm - psalm.xml
- puppet - .puppet
- rmt - .rmt.yml
- robo - RoboFile.php
- scrutinizer - .scrutinizer.yml
- semantic versioning - .semver
- SPIP - paquet.xml
- stickler - .stickler.yml

- storyplayer - storyplayer.json.dist
- styleci - .styleci.yml
- stylelint - .stylelintrc
- sublimelinter - .csslintrc
- svn - svn.revision, .svn, .svnignore
- transifex - .tx
- Robots.txt - robots.txt
- travis - .travis.yml, .env.travis, .travis, .travis.php.ini, .travis.coverage.sh, .travis.ini
- varci - .varci, .varci.yml
- Vagrant - Vagrantfile
- visualstudio - .vscode
- webpack - webpack.mix.js, webpack.config.js
- yarn - yarn.lock
- Zend_Tool - zfproject.xml

18.10 External links

List of external links mentionned in this documentation.

- #QuandLeDevALaFleme
- $_ENV
- $GLOBALS
- $HTTP_RAW_POST_DATA variable
- *.exakat.ini* or *.exakat.yaml* file. See Add Exakat To Your CI Pipeline
- *.phar* from the exakat.io website : www.exakat.io
- 1003.1-2008 - IEEE Standard for Information Technology - Portable Operating System Interface (POSIX(R))
- 7z
- @deprecated
- [blog] array_column()
- [CVE-2017-6090]
- [HttpFoundation] Make sessions secure and lazy #24523
- __autoload
- __set
- A PHP extension for Redis
- About circular references in PHP
- Add array_key_exists to the list of specialy compiled functions
- Allow a trailing comma in function calls

- Alpine Linux
- Alternative PHP Cache
- Alternative syntax
- Anonymous functions
- APCU
- Argon2 Password Hash
- Arithmetic Operators
- Aronduby Dump
- Array
- array
- Array Functions
- array_fill_keys
- array_filter
- array_key_exists() with objects
- array_map
- array_merge
- array_search
- array_slice
- array_unique
- ArrayAccess
- Arrays
- Arrays syntax
- Arrow functions
- assert
- Assignation Operators
- Autoloading Classe
- Autoloading Classes
- Avoid Else, Return Early
- Avoid nesting too deeply and return early (part 1)
- Avoid option arrays in constructors
- Avoid optional services as much as possible
- Backward incompatible changes
- Backward incompatible changes PHP 7.0
- basename
- Bazaar
- BC Math Functions

- Benoit Burnichon
- Bitmask Constant Arguments in PHP
- Bitwise Operators
- Brandon Savage
- browscap
- Bug #50887 preg_match , last optional sub-patterns ignored when empty
- Bzip2 Functions
- Cairo Graphics Library
- Calendar Functions
- Callback / callable
- Can you spot the vulnerability? (openssl_verify)
- Cant Use Return Value In Write Context
- cat: write error: Broken pipe
- Change the precedence of the concatenation operator
- Changes to variable handling
- Class Abstraction
- Class Constant
- Class Constants
- class_alias
- Classes abstraction
- Classes Abstraction
- Closure class
- Closure::bind
- Cmark
- Codeigniter
- COM and .Net (Windows)
- compact
- Comparison Operators
- composer
- Concrete 5
- Conflict resolution
- Constant definition
- Constant Scalar Expressions
- constant()
- Constants
- Constructors and Destructors

- Cookies
- count
- Courier Anti-pattern
- Covariant Returns and Contravariant Parameters
- crc32()
- Cross-Site Scripting (XSS)
- crypt
- Cryptography Extensions
- CSPRNG
- Ctype funtions
- curl
- Curl for PHP
- curl_version
- CVS
- CWE-484: Omitted Break Statement in Switch
- CWE-625: Permissive Regular Expression
- Cyrus
- d3.js
- Data filtering
- Data structures
- Database (dbm-style) Abstraction Layer
- Date and Time
- DCDFLIB
- Dead Code: Unused Method
- Declare
- declare
- define
- Dependency Injection Smells
- Deprecate and remove continue targeting switch
- Deprecate and remove INTL_IDNA_VARIANT_2003
- Deprecate curly brace syntax
- Deprecated features in PHP 5.4.x
- Deprecated features in PHP 5.5.x
- Deprecated features in PHP 7.2.x
- Deprecation allow_url_include
- Deprecations for PHP 7.2

- Deprecations for PHP 7.4
- Destructor
- DIO
- Dir predefined constants
- directive error_reporting
- Directly calling __clone is allowed
- dirname
- dist.exakat.io
- dl
- Do your objects talk to strangers?
- Docker
- Docker image
- Document Object Model
- Don't pass this out of a constructor
- Don't turn off CURLOPT_SSL_VERIFYPEER, fix your PHP configuration
- dotdeb instruction
- Double quoted
- download
- Drupal
- Dynamically Access PHP Object Properties with $this
- E_WARNING for invalid container read array-access
- Eaccelerator
- elseif/else if
- empty
- Empty Catch Clause
- Empty interfaces are bad practice
- empty()
- Enchant spelling library
- Ereg
- Error reporting
- Escape sequences
- Ev
- eval
- Event
- Exakat
- Exakat cloud

- Exakat SAS
- exakat/exakat
- Exception::__construct
- Exceptions
- Exchangeable image information
- Execution Operators
- EXP30-C. Do not depend on the order of evaluation for side effects
- expect
- explode
- ext-async
- ext-http
- ext/ast
- ext/gender manual
- ext/hash extension
- ext/hrtime manual
- ext/inotify manual
- ext/leveldb on Github
- ext/lua manual
- ext/mbstring
- ext/memcached manual
- ext/OpenSSL
- ext/readline
- ext/recode
- ext/SeasLog on Github
- ext/sqlite
- ext/sqlite3
- ext/uopz
- ext/varnish
- ext/zookeeper
- Extension Apache
- extension FANN
- extension mcrypt
- extract
- Ez
- Factory (object-oriented programming)
- FAM

- FastCGI Process Manager
- FDF
- ffmpeg-php
- file_get_contents
- filesystem
- Filinfo
- Final Keyword
- Firebase / Interbase
- Flag Argument
- FlagArgument
- Floating point numbers
- Floats
- Fluent Interfaces in PHP
- foreach
- Foreign Function Interface
- Frederic Bouchery
- From assumptions to assertions
- FuelPHP
- Function arguments
- Functions
- Gearman on PHP
- Generalize support of negative string offsets
- Generator delegation via yield from
- Generator Syntax
- Generators overview
- GeoIP
- get_class
- Gettext
- Git
- Github Action
- Github.com/exakat/exakat
- global namespace
- GMP
- Gnupg Function for PHP
- Goto
- Gremlin server

- Group Use Declaration RFC
- GRPC
- Handling file uploads
- Hardening Your HTTP Security Headers
- hash
- HASH Message Digest Framework
- hash_algos
- hash_file
- Heredoc
- Holger Woltersdorf
- How to fix Headers already sent error in PHP
- How to pick bad function and variable names
- htmlentities
- htmlspecialchars
- https://hub.docker.com/r/exakat/exakat-ga
- https://www.exakat.io/
- https://www.exakat.io/versionss/index.php?file=latest
- IBM Db2
- Iconv
- iconv()
- ICU
- Ideal regex delimiters in PHP
- idn_to_ascii
- IERS
- igbinary
- IIS Administration
- Image Processing and GD
- Imagick for PHP
- IMAP
- Implement ZEND_ARRAY_KEY_EXISTS opcode to speed up array_key_exists()
- implode
- In a PHP5 class, when does a private constructor get called?
- in_array()
- include
- include_once
- Incrementing/Decrementing Operators

- Info Predefined Constants
- Insecure Transportation Security Protocol Supported (TLS 1.0)
- Instanceof
- Integer overflow
- Integer Syntax
- Integers
- Interfaces
- Internal Constructor Behavior
- Is it a bad practice to have multiple classes in the same file?
- Isset
- Isset Ternary
- It is the 31st again
- iterable pseudo-type
- Iterables
- Joomla
- json_decode
- Judy C library
- Kafka client for PHP
- Kerberos V
- Lapack
- Laravel
- Late Static Bindings
- libeio
- libevent
- libmongoc
- libuuid
- libxml
- Lightweight Directory Access Protocol
- list
- List of function aliases
- List of HTTP header fields
- List of HTTP status codes
- List of Keywords
- List of other reserved words
- List of TCP and UDP port numbers
- list() Reference Assignment

- Logical Expressions in C/C++. Mistakes Made by Professionals
- Logical Operators
- Loosening Reserved Word Restrictions
- lzf
- Magic Constants
- Magic Hashes
- Magic Method
- Magic Methods
- Magic methods
- mail
- Mail related functions
- Marco Pivetta tweet
- Math predefined constants
- Mathematical Functions
- mb_encoding_detect
- mb_str_split
- Mbstring
- mcrypt_create_iv()
- MD5
- Media Type
- Memcache on PHP
- mercurial
- Method overloading
- mhash
- Microsoft SQL Server
- Microsoft SQL Server Driver
- Ming (flash)
- MongoDB driver
- move_uploaded_file
- msgpack for PHP
- MySQL Improved Extension
- mysqli
- Named Arguments
- Ncurses Terminal Screen Control
- Negative architecture, and assumptions about code
- Nested Ternaries are Great

- Net SNMP
- net_get_interfaces
- New Classes and Interfaces
- New custom object serialization mechanism
- New features
- New global constants in 7.2
- New global constants in 7.4
- New object type
- Newt
- No Dangling Reference
- Nowdoc
- Null and True
- Null Coalescing Assignment Operator
- Null Coalescing Operator
- Null Object Pattern
- Nullable types
- Object Calisthenics, rule # 2
- Object Calisthenics, rule # 5
- Object cloning
- Object Inheritance
- Object Interfaces
- Object interfaces
- Objects and references
- ODBC (Unified)
- online
- OPcache functions
- opencensus
- OpennSSL [PHP manual]
- openssl_random_pseudo_byte
- Operator Precedence
- Operators Precedence
- Optimization: How I made my PHP code run 100 times faster
- Optimize array_unique()
- Option to make json_encode and json_decode throw exceptions on errors
- Oracle OCI8
- original idea

- Original MySQL API
- Output Buffering Control
- Overload
- pack
- Packagist
- parent
- Parsekit
- Parsing and Lexing
- Passing arguments by reference
- Passing by reference
- Password hashing
- Password Hashing
- Pattern Modifiers
- PCOV
- PCRE
- PEAR
- pecl crypto
- PECL ext/xxtea
- pg_last_error
- Phalcon
- phar
- PHP - Fatal error: Unsupported operand types [duplicate]
- PHP 7 performance improvements (3/5): Encapsed strings optimization
- PHP 7.0 Backward incompatible changes
- PHP 7.0 Removed Functions
- PHP 7.1 no longer converts string to arrays the first time a value is assigned with square bracket notation
- PHP 7.2's "switch" optimisations
- PHP 7.2's switch optimisations
- PHP 7.3 Removed Functions
- PHP 7.3 UPGRADE NOTES
- PHP 7.4 Removed Functions
- PHP 8: Constructor property promotion
- PHP
- PHP class name constant case sensitivity and PSR-11
- PHP Classes containing only constants
- PHP Clone and Shallow vs Deep Copying

- PHP Constants
- PHP Data Object
- PHP Decimal
- PHP extension for libsodium
- PHP gmagick
- PHP Options And Information
- PHP Options/Info Functions
- PHP return(value); vs return value;
- PHP RFC: Add Stringable interface
- PHP RFC: Allow a trailing comma in function calls
- PHP RFC: Allow abstract function override
- PHP RFC: Allow trailing comma in parameter list
- PHP RFC: Arrays starting with a negative index
- PHP RFC: Arrow Functions
- PHP RFC: Convert numeric keys in object/array casts
- PHP RFC: Deprecate and Remove Bareword (Unquoted) Strings
- PHP RFC: Deprecate left-associative ternary operator
- PHP RFC: Deprecations for PHP 7.2 : Each()
- PHP RFC: Deprecations for PHP 7.4
- PHP RFC: is_countable
- PHP RFC: Nullsafe operator
- PHP RFC: Numeric Literal Separator
- PHP RFC: Scalar Type Hints
- PHP RFC: Shorter Attribute Syntax
- PHP RFC: Syntax for variadic functions
- PHP RFC: Unicode Codepoint Escape Syntax
- PHP RFC: Union Types 2.0
- PHP RFC: Variable Syntax Tweaks
- PHP Tags
- PHP why pi() and M_PI
- PHP-cs-fixer
- php-ext-wasm
- php-vips-ext
- php-zbarcode
- PHP: When is /tmp not /tmp?
- phpsdl

- PHPUnit
- plantuml
- PMB
- PostgreSQL
- Predefined Constants
- Predefined Exceptions
- Predefined Variables
- preg_filter
- Prepare for PHP 7 error messages (part 3)
- printf
- Process Control
- proctitle
- Properties
- Property overloading
- Pspell
- PSR-11 : Dependency injection container
- PSR-13 : Link definition interface
- PSR-16 : Common Interface for Caching Libraries
- PSR-3 : Logger Interface
- PSR-3
- PSR-6 : Caching
- Putting glob to the test
- RabbitMQ AMQP client library
- rar
- Rar archiving
- RectorPHP
- References
- Reflection
- Reflection export() methods
- Regular Expressions (Perl-Compatible)
- resources
- return
- Return Inside Finally Block
- Return Type Declaration
- Returning values
- RFC 7159

- RFC 7230
- RFC 822 (MIME)
- RFC 959
- RFC : Arrow functions
- RFC Preload
- RFC: Return Type Declarations
- runkit
- Salted Password Hashing - Doing it Right
- SARB
- Scalar type declarations
- Scope Resolution Operator (::)
- Secure Hash Algorithms
- Semaphore, Shared Memory and IPC
- Session
- session_regenerateid()
- Sessions
- Set-Cookie
- set_error_handler
- setcookie
- setlocale
- shell_exec
- SimpleXML
- Single Function Exit Point
- SOAP
- Sockets
- Sphinx Client
- Spread Operator in Array Expression
- Spread Operator in Array Expression
- sqlite3
- SQLite3::escapeString
- SSH2 functions
- Standard PHP Library (SPL)
- Static Analysis Results Interchange Format (SARIF)
- Static Keyword
- str_contains
- Strict typing

- Stricter type checks for arithmetic/bitwise operators
- String functions
- Strings
- strip_tags
- strpos not working correctly
- strtr
- Structuring PHP Exceptions
- Subpatterns
- substr
- Suhosin.org
- Sun, iPlanet and Netscape servers on Sun Solaris
- Superglobals
- Supported PHP Extensions
- Supported Protocols and Wrappers
- SVM
- Svn
- Swoole
- Symfony
- Syntax
- Ternary Operator
- tetraweb/php
- The Basics
- The basics of Fluent interfaces in PHP
- The Closure Class
- The Definitive 2019 Guide to Cryptographic Key Sizes and Algorithm Recommendations
- The Linux NIS(YP)/NYS/NIS+ HOWTO
- The list function & practical uses of array destructuring in PHP
- The main PPA for PHP (7.4, 7.3, 7.2, 7.1, 7.0, 5.6)
- Throw Expression
- Throwable
- Tidy
- tokenizer
- tokyo_tyrant
- trader
- Trailing Comma In Closure Use List
- Trailing Commas In List Syntax

- Traits
- Traversable
- trigger_error
- trim
- Tutorial 1: Let’s learn by example
- Type array
- Type Casting
- Type Declaration
- Type declarations
- Type declarations
- Type Declarations
- Type hinting for interfaces
- Type Juggling
- Type juggling
- Type Juggling Authentication Bypass Vulnerability in CMS Made Simple
- Type Operators
- Typed Properties 2.0
- Typo3
- Unbinding $this from non-static closures
- Understanding Dependency Injection
- Unicode block
- Unicode spaces
- unserialize()
- unset
- Unset casting
- UPGRADING 7.3
- Use of Hardcoded IPv4 Addresses
- Using namespaces: Aliasing/Importing
- Using namespaces: fallback to global function/constant
- Using non-breakable spaces in test method names
- Using single characters for variable names in loops/exceptions
- Using static variables
- V8 Javascript Engine
- Vagrant file
- Variable basics
- Variable functions

- Variable Scope
- Variable scope
- Variable variables
- Variable-length argument lists
- Variables
- Visibility
- Vladimir Reznichenko
- Void functions
- Warn when counting non-countable types
- Wddx on PHP
- Weak references
- What are the best practices for catching and re-throwing exceptions?
- What’s all this ‘immutable date’ stuff, anyway?
- When to declare classes final
- Why 777 Folder Permissions are a Security Risk
- Why does PHP 5.2+ disallow abstract static class methods?
- Why, php? WHY???
- wikidiff2
- Wincache extension for PHP
- Wordpress
- xattr
- xcache
- Xdebug
- xdiff
- XHprof Documentation
- XML External Entity
- XML Parser
- XML-RPC
- xmlreader
- XMLWriter
- XSL extension
- YAML Ain’t Markup Language
- Yii
- Yoda Conditions
- Zend Monitor - PHP API
- ZeroMQ

- zip
- Zip
- Zlib

18.11 Ruleset configurations

INI configuration for built-in rulesets. Copy them in config/themes.ini, and make your owns.

24 rulesets detailled here :

- Analyze
- CI-checks
- ClassReview
- Coding Conventions
- CompatibilityPHP53
- CompatibilityPHP54
- CompatibilityPHP55
- CompatibilityPHP56
- CompatibilityPHP70
- CompatibilityPHP71
- CompatibilityPHP72
- CompatibilityPHP73
- CompatibilityPHP74
- CompatibilityPHP80
- Dead code
- LintButWontExec
- Performances
- Rector
- Security
- Semantics
- Suggestions
- Top10
- Typechecks
- php-cs-fixable

Analyze This ruleset centralizes a large number of classic trap and pitfalls when writing PHP. _______

```
[Analyze]
    analyzer[] = “Arrays/AmbiguousKeys”;
    analyzer[] = “Arrays/MultipleIdenticalKeys”;
```

```
analyzer[] = “Arrays/NoSpreadForHash”;
analyzer[] = “Arrays/NonConstantArray”;
analyzer[] = “Arrays/NullBoolean”;
analyzer[] = “Arrays/RandomlySortedLiterals”;
analyzer[] = “Arrays/TooManyDimensions”;
analyzer[] = “Classes/AbstractOrImplements”;
analyzer[] = “Classes/AbstractStatic”;
analyzer[] = “Classes/AccessPrivate”;
analyzer[] = “Classes/AccessProtected”;
analyzer[] = “Classes/AmbiguousStatic”;
analyzer[] = “Classes/AmbiguousVisibilities”;
analyzer[] = “Classes/AvoidOptionArrays”;
analyzer[] = “Classes/AvoidOptionalProperties”;
analyzer[] = “Classes/CantExtendFinal”;
analyzer[] = “Classes/CantInstantiateClass”;
analyzer[] = “Classes/CheckOnCallUsage”;
analyzer[] = “Classes/CitSameName”;
analyzer[] = “Classes/CloneWithNonObject”;
analyzer[] = “Classes/ConstantClass”;
analyzer[] = “Classes/CouldBeAbstractClass”;
analyzer[] = “Classes/CouldBeFinal”;
analyzer[] = “Classes/CouldBeStatic”;
analyzer[] = “Classes/CouldBeStringable”;
analyzer[] = “Classes/CyclicReferences”;
analyzer[] = “Classes/DependantAbstractClass”;
analyzer[] = “Classes/DifferentArgumentCounts”;
analyzer[] = “Classes/DirectCallToMagicMethod”;
analyzer[] = “Classes/DontSendThisInConstructor”;
analyzer[] = “Classes/DontUnsetProperties”;
analyzer[] = “Classes/EmptyClass”;
analyzer[] = “Classes/FinalByOcramius”;
analyzer[] = “Classes/HiddenNullable”;
analyzer[] = “Classes/ImplementIsForInterface”;
analyzer[] = “Classes/ImplementedMethodsArePublic”;
analyzer[] = “Classes/IncompatibleSignature”;
analyzer[] = “Classes/IncompatibleSignature74”;
analyzer[] = “Classes/InstantiatingAbstractClass”;
analyzer[] = “Classes/MakeDefault”;
analyzer[] = “Classes/MakeGlobalAProperty”;
analyzer[] = “Classes/MethodSignatureMustBeCompatible”;
analyzer[] = “Classes/MismatchProperties”;
analyzer[] = “Classes/MissingAbstractMethod”;
analyzer[] = “Classes/MultipleDeclarations”;
analyzer[] = “Classes/MultipleTraitOrInterface”;
analyzer[] = “Classes/NoMagicWithArray”;
analyzer[] = “Classes/NoPSSOutsideClass”;
analyzer[] = “Classes/NoParent”;
```

```
analyzer[] = “Classes/NoPublicAccess”;
analyzer[] = “Classes/NoSelfReferencingConstant”;
analyzer[] = “Classes/NonNullableSetters”;
analyzer[] = “Classes/NonPpp”;
analyzer[] = “Classes/NonStaticMethodsCalledStatic”;
analyzer[] = “Classes/OldStyleConstructor”;
analyzer[] = “Classes/OldStyleVar”;
analyzer[] = “Classes/ParentFirst”;
analyzer[] = “Classes/PropertyCouldBeLocal”;
analyzer[] = “Classes/PropertyNeverUsed”;
analyzer[] = “Classes/PropertyUsedInOneMethodOnly”;
analyzer[] = “Classes/PssWithoutClass”;
analyzer[] = “Classes/RedefinedConstants”;
analyzer[] = “Classes/RedefinedDefault”;
analyzer[] = “Classes/RedefinedPrivateProperty”;
analyzer[] = “Classes/ScalarOrObjectProperty”;
analyzer[] = “Classes/ShouldUseSelf”;
analyzer[] = “Classes/ShouldUseThis”;
analyzer[] = “Classes/StaticContainsThis”;
analyzer[] = “Classes/StaticMethodsCalledFromObject”;
analyzer[] = “Classes/SwappedArguments”;
analyzer[] = “Classes/ThisIsForClasses”;
analyzer[] = “Classes/ThisIsNotAnArray”;
analyzer[] = “Classes/ThisIsNotForStatic”;
analyzer[] = “Classes/ThrowInDestruct”;
analyzer[] = “Classes/TooManyDereferencing”;
analyzer[] = “Classes/TooManyFinds”;
analyzer[] = “Classes/TooManyInjections”;
analyzer[] = “Classes/UndeclaredStaticProperty”;
analyzer[] = “Classes/UndefinedClasses”;
analyzer[] = “Classes/UndefinedConstants”;
analyzer[] = “Classes/UndefinedParentMP”;
analyzer[] = “Classes/UndefinedProperty”;
analyzer[] = “Classes/UndefinedStaticMP”;
analyzer[] = “Classes/UndefinedStaticclass”;
analyzer[] = “Classes/UnresolvedClasses”;
analyzer[] = “Classes/UnresolvedInstanceof”;
analyzer[] = “Classes/UnusedClass”;
analyzer[] = “Classes/UnusedConstant”;
analyzer[] = “Classes/UseClassOperator”;
analyzer[] = “Classes/UseInstanceof”;
analyzer[] = “Classes/UsedOnceProperty”;
analyzer[] = “Classes/UselessAbstract”;
analyzer[] = “Classes/UselessConstructor”;
analyzer[] = “Classes/UselessFinal”;
analyzer[] = “Classes/UsingThisOutsideAClass”;
analyzer[] = “Classes/WeakType”;
```

```
analyzer[] = “Classes/WrongName”;
analyzer[] = “Classes/WrongTypedPropertyInit”;
analyzer[] = “Constants/BadConstantnames”;
analyzer[] = “Constants/ConstRecommended”;
analyzer[] = “Constants/ConstantStrangeNames”;
analyzer[] = “Constants/CreatedOutsideItsNamespace”;
analyzer[] = “Constants/InvalidName”;
analyzer[] = “Constants/MultipleConstantDefinition”;
analyzer[] = “Constants/StrangeName”;
analyzer[] = “Constants/UndefinedConstants”;
analyzer[] = “Exceptions/CantThrow”;
analyzer[] = “Exceptions/CatchUndefinedVariable”;
analyzer[] = “Exceptions/ForgottenThrown”;
analyzer[] = “Exceptions/OverwriteException”;
analyzer[] = “Exceptions/ThrowFunctioncall”;
analyzer[] = “Exceptions/UncaughtExceptions”;
analyzer[] = “Exceptions/Unthrown”;
analyzer[] = “Exceptions/UselessCatch”;
analyzer[] = “Files/InclusionWrongCase”;
analyzer[] = “Files/MissingInclude”;
analyzer[] = “Functions/AliasesUsage”;
analyzer[] = “Functions/AvoidBooleanArgument”;
analyzer[] = “Functions/CallbackNeedsReturn”;
analyzer[] = “Functions/CouldCentralize”;
analyzer[] = “Functions/DeepDefinitions”;
analyzer[] = “Functions/DontUseVoid”;
analyzer[] = “Functions/EmptyFunction”;
analyzer[] = “Functions/FnArgumentVariableConfusion”;
analyzer[] = “Functions/HardcodedPasswords”;
analyzer[] = “Functions/InsufficientTypehint”;
analyzer[] = “Functions/MismatchParameterAndType”;
analyzer[] = “Functions/MismatchParameterName”;
analyzer[] = “Functions/MismatchTypeAndDefault”;
analyzer[] = “Functions/MismatchedDefaultArguments”;
analyzer[] = “Functions/MismatchedTypehint”;
analyzer[] = “Functions/ModifyTypedParameter”;
analyzer[] = “Functions/MustReturn”;
analyzer[] = “Functions/NeverUsedParameter”;
analyzer[] = “Functions/NoBooleanAsDefault”;
analyzer[] = “Functions/NoLiteralForReference”;
analyzer[] = “Functions/NoReturnUsed”;
analyzer[] = “Functions/OnlyVariableForReference”;
analyzer[] = “Functions/OnlyVariablePassedByReference”;
analyzer[] = “Functions/RedeclaredPhpFunction”;
analyzer[] = “Functions/RelayFunction”;
analyzer[] = “Functions/ShouldUseConstants”;
analyzer[] = “Functions/ShouldYieldWithKey”;
```

```
analyzer[] = “Functions/TooManyLocalVariables”;
analyzer[] = “Functions/TypehintMustBeReturned”;
analyzer[] = “Functions/TypehintedReferences”;
analyzer[] = “Functions/UndefinedFunctions”;
analyzer[] = “Functions/UnknownParameterName”;
analyzer[] = “Functions/UnusedArguments”;
analyzer[] = “Functions/UnusedInheritedVariable”;
analyzer[] = “Functions/UnusedReturnedValue”;
analyzer[] = “Functions/UseConstantAsArguments”;
analyzer[] = “Functions/UselessReferenceArgument”;
analyzer[] = “Functions/UselessReturn”;
analyzer[] = “Functions/UsesDefaultArguments”;
analyzer[] = “Functions/UsingDeprecated”;
analyzer[] = “Functions/WithoutReturn”;
analyzer[] = “Functions/WrongArgumentType”;
analyzer[] = “Functions/WrongNumberOfArguments”;
analyzer[] = “Functions/WrongOptionalParameter”;
analyzer[] = “Functions/WrongReturnedType”;
analyzer[] = “Functions/WrongTypeWithCall”;
analyzer[] = “Functions/funcGetArgModified”;
analyzer[] = “Interfaces/AlreadyParentsInterface”;
analyzer[] = “Interfaces/CantImplementTraversable”;
analyzer[] = “Interfaces/ConcreteVisibility”;
analyzer[] = “Interfaces/CouldUseInterface”;
analyzer[] = “Interfaces/EmptyInterface”;
analyzer[] = “Interfaces/IsNotImplemented”;
analyzer[] = “Interfaces/NoGaranteeForPropertyConstant”;
analyzer[] = “Interfaces/RepeatedInterface”;
analyzer[] = “Interfaces/UndefinedInterfaces”;
analyzer[] = “Interfaces/UselessInterfaces”;
analyzer[] = “Namespaces/ConstantFullyQualified”;
analyzer[] = “Namespaces/EmptyNamespace”;
analyzer[] = “Namespaces/HiddenUse”;
analyzer[] = “Namespaces/MultipleAliasDefinitionPerFile”;
analyzer[] = “Namespaces/MultipleAliasDefinitions”;
analyzer[] = “Namespaces/ShouldMakeAlias”;
analyzer[] = “Namespaces/UnresolvedUse”;
analyzer[] = “Namespaces/UseWithFullyQualifiedNS”;
analyzer[] = “Performances/ArrayMergeInLoops”;
analyzer[] = “Performances/LogicalToInArray”;
analyzer[] = “Performances/MemoizeMagicCall”;
analyzer[] = “Performances/PrePostIncrement”;
analyzer[] = “Performances/StrposTooMuch”;
analyzer[] = “Performances/UseArraySlice”;
analyzer[] = “Php/ArrayKeyExistsWithObjects”;
analyzer[] = “Php/AssertFunctionIsReserved”;
analyzer[] = “Php/AssignAnd”;
```

```
analyzer[] = “Php/Assumptions”;
analyzer[] = “Php/AvoidMbDectectEncoding”;
analyzer[] = “Php/BetterRand”;
analyzer[] = “Php/ConcatAndAddition”;
analyzer[] = “Php/Crc32MightBeNegative”;
analyzer[] = “Php/Deprecated”;
analyzer[] = “Php/DontPolluteGlobalSpace”;
analyzer[] = “Php/EmptyList”;
analyzer[] = “Php/FopenMode”;
analyzer[] = “Php/ForeachObject”;
analyzer[] = “Php/HashAlgos”;
analyzer[] = “Php/Incompilable”;
analyzer[] = “Php/InternalParameterType”;
analyzer[] = “Php/IsAWithString”;
analyzer[] = “Php/IsnullVsEqualNull”;
analyzer[] = “Php/LogicalInLetters”;
analyzer[] = “Php/MissingSubpattern”;
analyzer[] = “Php/MultipleDeclareStrict”;
analyzer[] = “Php/MustCallParentConstructor”;
analyzer[] = “Php/NoClassInGlobal”;
analyzer[] = “Php/NoReferenceForTernary”;
analyzer[] = “Php/PathinfoReturns”;
analyzer[] = “Php/ReservedNames”;
analyzer[] = “Php/ScalarAreNotArrays”;
analyzer[] = “Php/ShortOpenTagRequired”;
analyzer[] = “Php/ShouldUseCoalesce”;
analyzer[] = “Php/StrtrArguments”;
analyzer[] = “Php/TooManyNativeCalls”;
analyzer[] = “Php/UnknownPcre2Option”;
analyzer[] = “Php/UseObjectApi”;
analyzer[] = “Php/UsePathinfo”;
analyzer[] = “Php/UseSetCookie”;
analyzer[] = “Php/UseStdclass”;
analyzer[] = “Php/WrongTypeForNativeFunction”;
analyzer[] = “Php/oldAutoloadUsage”;
analyzer[] = “Security/DontEchoError”;
analyzer[] = “Security/ShouldUsePreparedStatement”;
analyzer[] = “Structures/AddZero”;
analyzer[] = “Structures/AlteringForeachWithoutReference”;
analyzer[] = “Structures/AlternativeConsistenceByFile”;
analyzer[] = “Structures/AlwaysFalse”;
analyzer[] = “Structures/ArrayFillWithObjects”;
analyzer[] = “Structures/ArrayMergeAndVariadic”;
analyzer[] = “Structures/ArrayMergeArrayArray”;
analyzer[] = “Structures/AssigneAndCompare”;
analyzer[] = “Structures/AutoUnsetForeach”;
analyzer[] = “Structures/BailOutEarly”;
```

```
analyzer[] = “Structures/BooleanStrictComparison”;
analyzer[] = “Structures/BreakOutsideLoop”;
analyzer[] = “Structures/BuriedAssignation”;
analyzer[] = “Structures/CastToBoolean”;
analyzer[] = “Structures/CastingTernary”;
analyzer[] = “Structures/CatchShadowsVariable”;
analyzer[] = “Structures/CheckAllTypes”;
analyzer[] = “Structures/CheckJson”;
analyzer[] = “Structures/CoalesceAndConcat”;
analyzer[] = “Structures/CommonAlternatives”;
analyzer[] = “Structures/ComparedComparison”;
analyzer[] = “Structures/ConcatEmpty”;
analyzer[] = “Structures/ContinueIsForLoop”;
analyzer[] = “Structures/CouldBeElse”;
analyzer[] = “Structures/CouldBeStatic”;
analyzer[] = “Structures/CouldUseDir”;
analyzer[] = “Structures/CouldUseShortAssignation”;
analyzer[] = “Structures/CouldUseStrrepeat”;
analyzer[] = “Structures/DanglingArrayReferences”;
analyzer[] = “Structures/DirThenSlash”;
analyzer[] = “Structures/DontChangeBlindKey”;
analyzer[] = “Structures/DontMixPlusPlus”;
analyzer[] = “Structures/DontReadAndWriteInOneExpression”;
analyzer[] = “Structures/DoubleAssignation”;
analyzer[] = “Structures/DoubleInstruction”;
analyzer[] = “Structures/DoubleObjectAssignation”;
analyzer[] = “Structures/DropElseAfterReturn”;
analyzer[] = “Structures/EchoWithConcat”;
analyzer[] = “Structures/ElseIfElseif”;
analyzer[] = “Structures/EmptyBlocks”;
analyzer[] = “Structures/EmptyLines”;
analyzer[] = “Structures/EmptyTryCatch”;
analyzer[] = “Structures/ErrorReportingWithInteger”;
analyzer[] = “Structures/EvalUsage”;
analyzer[] = “Structures/EvalWithoutTry”;
analyzer[] = “Structures/ExitUsage”;
analyzer[] = “Structures/FailingSubstrComparison”;
analyzer[] = “Structures/ForeachReferenceIsNotModified”;
analyzer[] = “Structures/ForeachSourceValue”;
analyzer[] = “Structures/ForgottenWhiteSpace”;
analyzer[] = “Structures/GlobalUsage”;
analyzer[] = “Structures/Htmlentitiescall”;
analyzer[] = “Structures/IdenticalConditions”;
analyzer[] = “Structures/IdenticalConsecutive”;
analyzer[] = “Structures/IdenticalOnBothSides”;
analyzer[] = “Structures/IfWithSameConditions”;
analyzer[] = “Structures/Iffectation”;
```

analyzer[] = “Structures/ImpliedIf”;
analyzer[] = “Structures/ImplodeArgsOrder”;
analyzer[] = “Structures/InconsistentElseif”;
analyzer[] = “Structures/IndicesAreIntOrString”;
analyzer[] = “Structures/InfiniteRecursion”;
analyzer[] = “Structures/InvalidPackFormat”;
analyzer[] = “Structures/InvalidRegex”;
analyzer[] = “Structures/IsZero”;
analyzer[] = “Structures/ListOmissions”;
analyzer[] = “Structures/LogicalMistakes”;
analyzer[] = “Structures/LoneBlock”;
analyzer[] = “Structures/LongArguments”;
analyzer[] = “Structures/MaxLevelOfIdentation”;
analyzer[] = “Structures/MbstringThirdArg”;
analyzer[] = “Structures/MbstringUnknownEncoding”;
analyzer[] = “Structures/MergeIfThen”;
analyzer[] = “Structures/MismatchedTernary”;
analyzer[] = “Structures/MissingCases”;
analyzer[] = “Structures/MissingNew”;
analyzer[] = “Structures/MissingParenthesis”;
analyzer[] = “Structures/MixedConcatInterpolation”;
analyzer[] = “Structures/ModernEmpty”;
analyzer[] = “Structures/MultipleDefinedCase”;
analyzer[] = “Structures/MultipleTypeVariable”;
analyzer[] = “Structures/MultiplyByOne”;
analyzer[] = “Structures/NegativePow”;
analyzer[] = “Structures/NestedIfthen”;
analyzer[] = “Structures/NestedTernary”;
analyzer[] = “Structures/NeverNegative”;
analyzer[] = “Structures/NextMonthTrap”;
analyzer[] = “Structures/NoAppendOnSource”;
analyzer[] = “Structures/NoChangeIncomingVariables”;
analyzer[] = “Structures/NoChoice”;
analyzer[] = “Structures/NoDirectUsage”;
analyzer[] = “Structures/NoEmptyRegex”;
analyzer[] = “Structures/NoGetClassNull”;
analyzer[] = “Structures/NoHardcodedHash”;
analyzer[] = “Structures/NoHardcodedIp”;
analyzer[] = “Structures/NoHardcodedPath”;
analyzer[] = “Structures/NoHardcodedPort”;
analyzer[] = “Structures/NoIssetWithEmpty”;
analyzer[] = “Structures/NoNeedForElse”;
analyzer[] = “Structures/NoNeedForTriple”;
analyzer[] = “Structures/NoParenthesisForLanguageConstruct”;
analyzer[] = “Structures/NoReferenceOnLeft”;
analyzer[] = “Structures/NoSubstrOne”;
analyzer[] = “Structures/NoVariableIsACondition”;

analyzer[] = “Structures/Noscream”;
analyzer[] = “Structures/NotEqual”;
analyzer[] = “Structures/NotNot”;
analyzer[] = “Structures/ObjectReferences”;
analyzer[] = “Structures/OnceUsage”;
analyzer[] = “Structures/OneLineTwoInstructions”;
analyzer[] = “Structures/OnlyVariableReturnedByReference”;
analyzer[] = “Structures/OrDie”;
analyzer[] = “Structures/PossibleInfiniteLoop”;
analyzer[] = “Structures/PrintAndDie”;
analyzer[] = “Structures/PrintWithoutParenthesis”;
analyzer[] = “Structures/PrintfArguments”;
analyzer[] = “Structures/QueriesInLoop”;
analyzer[] = “Structures/RepeatedPrint”;
analyzer[] = “Structures/RepeatedRegex”;
analyzer[] = “Structures/ResultMayBeMissing”;
analyzer[] = “Structures/ReturnTrueFalse”;
analyzer[] = “Structures/SameConditions”;
analyzer[] = “Structures/ShouldChainException”;
analyzer[] = “Structures/ShouldMakeTernary”;
analyzer[] = “Structures/ShouldPreprocess”;
analyzer[] = “Structures/ShouldUseExplodeArgs”;
analyzer[] = “Structures/StaticLoop”;
analyzer[] = “Structures/StripTagsSkipsClosedTag”;
analyzer[] = “Structures/StrposCompare”;
analyzer[] = “Structures/SuspiciousComparison”;
analyzer[] = “Structures/SwitchToSwitch”;
analyzer[] = “Structures/SwitchWithoutDefault”;
analyzer[] = “Structures/TernaryInConcat”;
analyzer[] = “Structures/TestThenCast”;
analyzer[] = “Structures/ThrowsAndAssign”;
analyzer[] = “Structures/TimestampDifference”;
analyzer[] = “Structures/UncheckedResources”;
analyzer[] = “Structures/UnconditionLoopBreak”;
analyzer[] = “Structures/UnknownPregOption”;
analyzer[] = “Structures/Unpreprocessed”;
analyzer[] = “Structures/UnsetInForeach”;
analyzer[] = “Structures/UnsupportedTypesWithOperators”;
analyzer[] = “Structures/UnusedGlobal”;
analyzer[] = “Structures/UseConstant”;
analyzer[] = “Structures/UseInstanceof”;
analyzer[] = “Structures/UsePositiveCondition”;
analyzer[] = “Structures/UseSystemTmp”;
analyzer[] = “Structures/UselessBrackets”;
analyzer[] = “Structures/UselessCasting”;
analyzer[] = “Structures/UselessCheck”;
analyzer[] = “Structures/UselessGlobal”;

```
    analyzer[] = “Structures/UselessInstruction”;
    analyzer[] = “Structures/UselessParenthesis”;
    analyzer[] = “Structures/UselessSwitch”;
    analyzer[] = “Structures/UselessUnset”;
    analyzer[] = “Structures/VardumpUsage”;
    analyzer[] = “Structures/WhileListEach”;
    analyzer[] = “Structures/WrongRange”;
    analyzer[] = “Structures/pregOptionE”;
    analyzer[] = “Structures/toStringThrowsException”;
    analyzer[] = “Traits/AlreadyParentsTrait”;
    analyzer[] = “Traits/DependantTrait”;
    analyzer[] = “Traits/EmptyTrait”;
    analyzer[] = “Traits/MethodCollisionTraits”;
    analyzer[] = “Traits/TraitNotFound”;
    analyzer[] = “Traits/UndefinedInsteadof”;
    analyzer[] = “Traits/UndefinedTrait”;
    analyzer[] = “Traits/UselessAlias”;
    analyzer[] = “Type/NoRealComparison”;
    analyzer[] = “Type/OneVariableStrings”;
    analyzer[] = “Type/ShouldTypecast”;
    analyzer[] = “Type/SilentlyCastInteger”;
    analyzer[] = “Type/StringHoldAVariable”;
    analyzer[] = “Type/StringWithStrangeSpace”;
    analyzer[] = “Typehints/MissingReturntype”;
    analyzer[] = “Variables/AssignedTwiceOrMore”;
    analyzer[] = “Variables/LostReferences”;
    analyzer[] = “Variables/OverwrittenLiterals”;
    analyzer[] = “Variables/StrangeName”;
    analyzer[] = “Variables/UndefinedConstantName”;
    analyzer[] = “Variables/UndefinedVariable”;
    analyzer[] = “Variables/VariableNonascii”;
    analyzer[] = “Variables/VariableUsedOnce”;
    analyzer[] = “Variables/VariableUsedOnceByContext”;
    analyzer[] = “Variables/WrittenOnlyVariable”;|
```

CI-checks This ruleset is a collection of important rules to run in a CI pipeline. _________

```
[CI-checks]
    analyzer[] = “Arrays/MultipleIdenticalKeys”;
    analyzer[] = “Classes/CheckOnCallUsage”;
    analyzer[] = “Classes/ConstantClass”;
    analyzer[] = “Classes/DirectCallToMagicMethod”;
    analyzer[] = “Classes/DontUnsetProperties”;
    analyzer[] = “Classes/MultipleDeclarations”;
    analyzer[] = “Classes/MultipleTraitOrInterface”;
    analyzer[] = “Classes/NoMagicWithArray”;
```

```
analyzer[] = “Classes/NoParent”;
analyzer[] = “Classes/NonPpp”;
analyzer[] = “Classes/NonStaticMethodsCalledStatic”;
analyzer[] = “Classes/RedefinedConstants”;
analyzer[] = “Classes/RedefinedDefault”;
analyzer[] = “Classes/StaticContainsThis”;
analyzer[] = “Classes/StaticMethodsCalledFromObject”;
analyzer[] = “Classes/ThrowInDestruct”;
analyzer[] = “Classes/UndeclaredStaticProperty”;
analyzer[] = “Classes/UndefinedConstants”;
analyzer[] = “Classes/UndefinedProperty”;
analyzer[] = “Classes/UndefinedStaticclass”;
analyzer[] = “Classes/UseClassOperator”;
analyzer[] = “Classes/UseInstanceof”;
analyzer[] = “Classes/UselessFinal”;
analyzer[] = “Classes/WrongTypedPropertyInit”;
analyzer[] = “Constants/ConstRecommended”;
analyzer[] = “Constants/ConstantStrangeNames”;
analyzer[] = “Constants/MultipleConstantDefinition”;
analyzer[] = “Constants/UndefinedConstants”;
analyzer[] = “Exceptions/OverwriteException”;
analyzer[] = “Exceptions/ThrowFunctioncall”;
analyzer[] = “Exceptions/UselessCatch”;
analyzer[] = “Functions/AliasesUsage”;
analyzer[] = “Functions/CallbackNeedsReturn”;
analyzer[] = “Functions/MustReturn”;
analyzer[] = “Functions/NoLiteralForReference”;
analyzer[] = “Functions/RedeclaredPhpFunction”;
analyzer[] = “Functions/ShouldYieldWithKey”;
analyzer[] = “Functions/TypehintMustBeReturned”;
analyzer[] = “Functions/TypehintedReferences”;
analyzer[] = “Functions/UndefinedFunctions”;
analyzer[] = “Functions/UnknownParameterName”;
analyzer[] = “Functions/UnusedInheritedVariable”;
analyzer[] = “Functions/UseConstantAsArguments”;
analyzer[] = “Functions/UsesDefaultArguments”;
analyzer[] = “Functions/WrongNumberOfArguments”;
analyzer[] = “Functions/WrongOptionalParameter”;
analyzer[] = “Functions/WrongReturnedType”;
analyzer[] = “Functions/WrongTypeWithCall”;
analyzer[] = “Interfaces/CantImplementTraversable”;
analyzer[] = “Interfaces/IsNotImplemented”;
analyzer[] = “Interfaces/UndefinedInterfaces”;
analyzer[] = “Namespaces/EmptyNamespace”;
analyzer[] = “Namespaces/HiddenUse”;
analyzer[] = “Namespaces/MultipleAliasDefinitionPerFile”;
analyzer[] = “Namespaces/MultipleAliasDefinitions”;
```

```
analyzer[] = “Namespaces/ShouldMakeAlias”;
analyzer[] = “Performances/ArrayMergeInLoops”;
analyzer[] = “Performances/PrePostIncrement”;
analyzer[] = “Performances/StrposTooMuch”;
analyzer[] = “Performances/UseArraySlice”;
analyzer[] = “Php/AssignAnd”;
analyzer[] = “Php/BetterRand”;
analyzer[] = “Php/ConcatAndAddition”;
analyzer[] = “Php/Deprecated”;
analyzer[] = “Php/FopenMode”;
analyzer[] = “Php/InternalParameterType”;
analyzer[] = “Php/IsAWithString”;
analyzer[] = “Php/IsnullVsEqualNull”;
analyzer[] = “Php/LogicalInLetters”;
analyzer[] = “Php/MissingSubpattern”;
analyzer[] = “Php/NoClassInGlobal”;
analyzer[] = “Php/NoReferenceForTernary”;
analyzer[] = “Php/ScalarAreNotArrays”;
analyzer[] = “Php/ShouldUseCoalesce”;
analyzer[] = “Php/StrtrArguments”;
analyzer[] = “Php/UseObjectApi”;
analyzer[] = “Php/UsePathinfo”;
analyzer[] = “Php/WrongTypeForNativeFunction”;
analyzer[] = “Security/DontEchoError”;
analyzer[] = “Security/ShouldUsePreparedStatement”;
analyzer[] = “Structures/AddZero”;
analyzer[] = “Structures/AlteringForeachWithoutReference”;
analyzer[] = “Structures/AssigneAndCompare”;
analyzer[] = “Structures/AutoUnsetForeach”;
analyzer[] = “Structures/BooleanStrictComparison”;
analyzer[] = “Structures/CastingTernary”;
analyzer[] = “Structures/CheckJson”;
analyzer[] = “Structures/CoalesceAndConcat”;
analyzer[] = “Structures/CouldUseDir”;
analyzer[] = “Structures/CouldUseShortAssignation”;
analyzer[] = “Structures/CouldUseStrrepeat”;
analyzer[] = “Structures/DanglingArrayReferences”;
analyzer[] = “Structures/DirThenSlash”;
analyzer[] = “Structures/DropElseAfterReturn”;
analyzer[] = “Structures/ElseIfElseif”;
analyzer[] = “Structures/EmptyBlocks”;
analyzer[] = “Structures/ErrorReportingWithInteger”;
analyzer[] = “Structures/EvalWithoutTry”;
analyzer[] = “Structures/ExitUsage”;
analyzer[] = “Structures/FailingSubstrComparison”;
analyzer[] = “Structures/ForeachReferenceIsNotModified”;
analyzer[] = “Structures/ForgottenWhiteSpace”;
```

```
analyzer[] = “Structures/Htmlentitiescall”;
analyzer[] = “Structures/IdenticalConditions”;
analyzer[] = “Structures/IdenticalOnBothSides”;
analyzer[] = “Structures/IfWithSameConditions”;
analyzer[] = “Structures/ImpliedIf”;
analyzer[] = “Structures/ImplodeArgsOrder”;
analyzer[] = “Structures/IndicesAreIntOrString”;
analyzer[] = “Structures/InvalidPackFormat”;
analyzer[] = “Structures/InvalidRegex”;
analyzer[] = “Structures/IsZero”;
analyzer[] = “Structures/ListOmissions”;
analyzer[] = “Structures/LogicalMistakes”;
analyzer[] = “Structures/LoneBlock”;
analyzer[] = “Structures/MbstringThirdArg”;
analyzer[] = “Structures/MbstringUnknownEncoding”;
analyzer[] = “Structures/MergeIfThen”;
analyzer[] = “Structures/MissingParenthesis”;
analyzer[] = “Structures/MultipleDefinedCase”;
analyzer[] = “Structures/MultiplyByOne”;
analyzer[] = “Structures/NegativePow”;
analyzer[] = “Structures/NestedTernary”;
analyzer[] = “Structures/NeverNegative”;
analyzer[] = “Structures/NextMonthTrap”;
analyzer[] = “Structures/NoChoice”;
analyzer[] = “Structures/NoEmptyRegex”;
analyzer[] = “Structures/NoIssetWithEmpty”;
analyzer[] = “Structures/NoParenthesisForLanguageConstruct”;
analyzer[] = “Structures/NoReferenceOnLeft”;
analyzer[] = “Structures/NoSubstrOne”;
analyzer[] = “Structures/Noscream”;
analyzer[] = “Structures/NotEqual”;
analyzer[] = “Structures/NotNot”;
analyzer[] = “Structures/ObjectReferences”;
analyzer[] = “Structures/OrDie”;
analyzer[] = “Structures/PrintAndDie”;
analyzer[] = “Structures/PrintWithoutParenthesis”;
analyzer[] = “Structures/PrintfArguments”;
analyzer[] = “Structures/RepeatedPrint”;
analyzer[] = “Structures/RepeatedRegex”;
analyzer[] = “Structures/ResultMayBeMissing”;
analyzer[] = “Structures/ReturnTrueFalse”;
analyzer[] = “Structures/SameConditions”;
analyzer[] = “Structures/ShouldChainException”;
analyzer[] = “Structures/ShouldMakeTernary”;
analyzer[] = “Structures/ShouldUseExplodeArgs”;
analyzer[] = “Structures/StripTagsSkipsClosedTag”;
analyzer[] = “Structures/StrposCompare”;
```

```
    analyzer[] = “Structures/SwitchWithoutDefault”;
    analyzer[] = “Structures/TernaryInConcat”;
    analyzer[] = “Structures/ThrowsAndAssign”;
    analyzer[] = “Structures/TimestampDifference”;
    analyzer[] = “Structures/UncheckedResources”;
    analyzer[] = “Structures/UnconditionLoopBreak”;
    analyzer[] = “Structures/UseConstant”;
    analyzer[] = “Structures/UseInstanceof”;
    analyzer[] = “Structures/UseSystemTmp”;
    analyzer[] = “Structures/UselessBrackets”;
    analyzer[] = “Structures/UselessCasting”;
    analyzer[] = “Structures/UselessCheck”;
    analyzer[] = “Structures/UselessInstruction”;
    analyzer[] = “Structures/UselessParenthesis”;
    analyzer[] = “Structures/UselessUnset”;
    analyzer[] = “Structures/VardumpUsage”;
    analyzer[] = “Structures/WhileListEach”;
    analyzer[] = “Structures/pregOptionE”;
    analyzer[] = “Traits/UndefinedInsteadof”;
    analyzer[] = “Traits/UndefinedTrait”;
    analyzer[] = “Traits/UselessAlias”;
    analyzer[] = “Type/NoRealComparison”;
    analyzer[] = “Type/OneVariableStrings”;
    analyzer[] = “Type/ShouldTypecast”;
    analyzer[] = “Type/SilentlyCastInteger”;
    analyzer[] = “Type/StringWithStrangeSpace”;
    analyzer[] = “Typehints/MissingReturntype”;
    analyzer[] = “Variables/UndefinedVariable”;|
```

ClassReview This ruleset focuses on classes construction issues, and their related structures : traits, interfaces, methods, properties, constants. ___________

```
[ClassReview]
    analyzer[] = “Classes/AvoidOptionArrays”;
    analyzer[] = “Classes/CancelCommonMethod”;
    analyzer[] = “Classes/CouldBeAbstractClass”;
    analyzer[] = “Classes/CouldBeClassConstant”;
    analyzer[] = “Classes/CouldBeFinal”;
    analyzer[] = “Classes/CouldBeParentMethod”;
    analyzer[] = “Classes/CouldBePrivate”;
    analyzer[] = “Classes/CouldBePrivateConstante”;
    analyzer[] = “Classes/CouldBePrivateMethod”;
    analyzer[] = “Classes/CouldBeProtectedConstant”;
    analyzer[] = “Classes/CouldBeProtectedMethod”;
    analyzer[] = “Classes/CouldBeProtectedProperty”;
    analyzer[] = “Classes/CouldBeStatic”;
    analyzer[] = “Classes/CyclicReferences”;
    analyzer[] = “Classes/DependantAbstractClass”;
    analyzer[] = “Classes/DifferentArgumentCounts”;
    analyzer[] = “Classes/DisconnectedClasses”;
    analyzer[] = “Classes/Finalclass”;
    analyzer[] = “Classes/Finalmethod”;
    analyzer[] = “Classes/FossilizedMethod”;
    analyzer[] = “Classes/HiddenNullable”;
    analyzer[] = “Classes/InsufficientPropertyTypehint”;
    analyzer[] = “Classes/MismatchProperties”;
    analyzer[] = “Classes/MissingAbstractMethod”;
    analyzer[] = “Classes/MutualExtension”;
    analyzer[] = “Classes/NoParent”;
    analyzer[] = “Classes/NoSelfReferencingConstant”;
    analyzer[] = “Classes/NonNullableSetters”;
    analyzer[] = “Classes/PropertyCouldBeLocal”;
    analyzer[] = “Classes/RaisedAccessLevel”;
    analyzer[] = “Classes/RedefinedProperty”;
    analyzer[] = “Classes/ShouldUseSelf”;
    analyzer[] = “Classes/UndeclaredStaticProperty”;
    analyzer[] = “Classes/UninitedProperty”;
    analyzer[] = “Classes/UnreachableConstant”;
    analyzer[] = “Classes/UnusedConstant”;
    analyzer[] = “Classes/UselessTypehint”;
    analyzer[] = “Classes/WrongTypedPropertyInit”;
    analyzer[] = “Functions/ExceedingTypehint”;
    analyzer[] = “Functions/ModifyTypedParameter”;
    analyzer[] = “Functions/NullableWithoutCheck”;
    analyzer[] = “Functions/WrongReturnedType”;
    analyzer[] = “Interfaces/AvoidSelfInInterface”;
    analyzer[] = “Interfaces/IsNotImplemented”;
    analyzer[] = “Interfaces/NoGaranteeForPropertyConstant”;
    analyzer[] = “Interfaces/UselessInterfaces”;
    analyzer[] = “Performances/MemoizeMagicCall”;
    analyzer[] = “Structures/CouldBeStatic”;
    analyzer[] = “Structures/DoubleObjectAssignation”;
    analyzer[] = “Traits/SelfUsingTrait”;
    analyzer[] = “Traits/UnusedClassTrait”;|
```

Coding Conventions This ruleset centralizes all analysis related to coding conventions. Sometimes, those are easy to extract with static analysis, and so here they are. No all o them are available.

```
[Coding Conventions]
    analyzer[] = “Arrays/EmptySlots”;
    analyzer[] = “Arrays/MistakenConcatenation”;
    analyzer[] = “Classes/MultipleClassesInFile”;
    analyzer[] = “Classes/OrderOfDeclaration”;
    analyzer[] = “Classes/WrongCase”;
    analyzer[] = “Constants/ConstRecommended”;
    analyzer[] = “Functions/OneLetterFunctions”;
    analyzer[] = “Functions/WrongCase”;
    analyzer[] = “Functions/WrongTypehintedName”;
    analyzer[] = “Namespaces/UseWithFullyQualifiedNS”;
    analyzer[] = “Namespaces/WrongCase”;
    analyzer[] = “Php/CloseTags”;
    analyzer[] = “Php/ReturnWithParenthesis”;
    analyzer[] = “Php/UpperCaseFunction”;
    analyzer[] = “Php/UpperCaseKeyword”;
    analyzer[] = “Structures/Bracketless”;
    analyzer[] = “Structures/ConstantComparisonConsistance”;
    analyzer[] = “Structures/DontBeTooManual”;
    analyzer[] = “Structures/EchoPrintConsistance”;
    analyzer[] = “Structures/HeredocDelimiterFavorite”;
    analyzer[] = “Structures/MixedConcatInterpolation”;
    analyzer[] = “Structures/PlusEgalOne”;
    analyzer[] = “Structures/YodaComparison”;
    analyzer[] = “Type/ShouldBeSingleQuote”;
    analyzer[] = “Type/SimilarIntegers”;
    analyzer[] = “Type/StringInterpolation”;
    analyzer[] = “Variables/VariableUppercase”;|
```

CompatibilityPHP53 This ruleset centralizes all analysis for the migration from PHP 5.2 to 5.3.

```
[CompatibilityPHP53]
    analyzer[] = “Arrays/ArrayNSUsage”;
    analyzer[] = “Arrays/MixedKeys”;
    analyzer[] = “Classes/Anonymous”;
    analyzer[] = “Classes/CantInheritAbstractMethod”;
    analyzer[] = “Classes/ChildRemoveTypehint”;
    analyzer[] = “Classes/ConstVisibilityUsage”;
    analyzer[] = “Classes/IntegerAsProperty”;
    analyzer[] = “Classes/NonStaticMethodsCalledStatic”;
    analyzer[] = “Classes/NullOnNew”;
    analyzer[] = “Exceptions/MultipleCatch”;
    analyzer[] = “Extensions/Extdba”;
    analyzer[] = “Extensions/Extfdf”;
    analyzer[] = “Extensions/Extming”;
    analyzer[] = “Functions/GeneratorCannotReturn”;
    analyzer[] = “Functions/MultipleSameArguments”;
    analyzer[] = “Namespaces/UseFunctionsConstants”;
    analyzer[] = “Php/CantUseReturnValueInWriteContext”;
    analyzer[] = “Php/CaseForPSS”;
```

```
analyzer[] = “Php/ClassConstWithArray”;
analyzer[] = “Php/ClosureThisSupport”;
analyzer[] = “Php/CoalesceEqual”;
analyzer[] = “Php/ConcatAndAddition”;
analyzer[] = “Php/ConstWithArray”;
analyzer[] = “Php/DefineWithArray”;
analyzer[] = “Php/DirectCallToClone”;
analyzer[] = “Php/EllipsisUsage”;
analyzer[] = “Php/ExponentUsage”;
analyzer[] = “Php/FlexibleHeredoc”;
analyzer[] = “Php/GroupUseDeclaration”;
analyzer[] = “Php/GroupUseTrailingComma”;
analyzer[] = “Php/HashAlgos53”;
analyzer[] = “Php/HashAlgos71”;
analyzer[] = “Php/ListShortSyntax”;
analyzer[] = “Php/ListWithKeys”;
analyzer[] = “Php/ListWithReference”;
analyzer[] = “Php/MethodCallOnNew”;
analyzer[] = “Php/NoListWithString”;
analyzer[] = “Php/NoReferenceForStaticProperty”;
analyzer[] = “Php/NoReturnForGenerator”;
analyzer[] = “Php/NoStringWithAppend”;
analyzer[] = “Php/NoSubstrMinusOne”;
analyzer[] = “Php/PHP70scalartypehints”;
analyzer[] = “Php/PHP71scalartypehints”;
analyzer[] = “Php/PHP72scalartypehints”;
analyzer[] = “Php/PHP73LastEmptyArgument”;
analyzer[] = “Php/ParenthesisAsParameter”;
analyzer[] = “Php/Php54NewFunctions”;
analyzer[] = “Php/Php55NewFunctions”;
analyzer[] = “Php/Php56NewFunctions”;
analyzer[] = “Php/Php70NewClasses”;
analyzer[] = “Php/Php70NewFunctions”;
analyzer[] = “Php/Php70NewInterfaces”;
analyzer[] = “Php/Php71NewClasses”;
analyzer[] = “Php/Php72NewClasses”;
analyzer[] = “Php/Php73NewFunctions”;
analyzer[] = “Php/Php7RelaxedKeyword”;
analyzer[] = “Php/StaticclassUsage”;
analyzer[] = “Php/TrailingComma”;
analyzer[] = “Php/TypedPropertyUsage”;
analyzer[] = “Php/UnicodeEscapePartial”;
analyzer[] = “Php/UnicodeEscapeSyntax”;
analyzer[] = “Php/UnpackingInsideArrays”;
analyzer[] = “Php/UseNullableType”;
analyzer[] = “Php/debugInfoUsage”;
analyzer[] = “Structures/Break0”;
```

```
    analyzer[] = “Structures/ConstantScalarExpression”;
    analyzer[] = “Structures/ContinueIsForLoop”;
    analyzer[] = “Structures/DereferencingAS”;
    analyzer[] = “Structures/ForeachWithList”;
    analyzer[] = “Structures/FunctionSubscripting”;
    analyzer[] = “Structures/IssetWithConstant”;
    analyzer[] = “Structures/NoGetClassNull”;
    analyzer[] = “Structures/PHP7Dirname”;
    analyzer[] = “Structures/SwitchWithMultipleDefault”;
    analyzer[] = “Structures/VariableGlobal”;
    analyzer[] = “Type/Binary”;
    analyzer[] = “Type/MalformedOctal”;
    analyzer[] = “Variables/Php5IndirectExpression”;
    analyzer[] = “Variables/Php7IndirectExpression”;|
```

CompatibilityPHP54 This ruleset centralizes all analysis for the migration from PHP 5.3 to 5.4.

```
[CompatibilityPHP54]
    analyzer[] = “Arrays/MixedKeys”;
    analyzer[] = “Classes/Anonymous”;
    analyzer[] = “Classes/CantInheritAbstractMethod”;
    analyzer[] = “Classes/ChildRemoveTypehint”;
    analyzer[] = “Classes/ConstVisibilityUsage”;
    analyzer[] = “Classes/IntegerAsProperty”;
    analyzer[] = “Classes/NonStaticMethodsCalledStatic”;
    analyzer[] = “Classes/NullOnNew”;
    analyzer[] = “Exceptions/MultipleCatch”;
    analyzer[] = “Extensions/Extmhash”;
    analyzer[] = “Functions/GeneratorCannotReturn”;
    analyzer[] = “Functions/MultipleSameArguments”;
    analyzer[] = “Namespaces/UseFunctionsConstants”;
    analyzer[] = “Php/CantUseReturnValueInWriteContext”;
    analyzer[] = “Php/CaseForPSS”;
    analyzer[] = “Php/ClassConstWithArray”;
    analyzer[] = “Php/CoalesceEqual”;
    analyzer[] = “Php/ConcatAndAddition”;
    analyzer[] = “Php/ConstWithArray”;
    analyzer[] = “Php/DefineWithArray”;
    analyzer[] = “Php/DirectCallToClone”;
    analyzer[] = “Php/EllipsisUsage”;
    analyzer[] = “Php/ExponentUsage”;
    analyzer[] = “Php/FlexibleHeredoc”;
    analyzer[] = “Php/GroupUseDeclaration”;
    analyzer[] = “Php/GroupUseTrailingComma”;
    analyzer[] = “Php/HashAlgos53”;
    analyzer[] = “Php/HashAlgos54”;
```

```
analyzer[] = “Php/HashAlgos71”;
analyzer[] = “Php/ListShortSyntax”;
analyzer[] = “Php/ListWithKeys”;
analyzer[] = “Php/ListWithReference”;
analyzer[] = “Php/NoListWithString”;
analyzer[] = “Php/NoReferenceForStaticProperty”;
analyzer[] = “Php/NoReturnForGenerator”;
analyzer[] = “Php/NoStringWithAppend”;
analyzer[] = “Php/NoSubstrMinusOne”;
analyzer[] = “Php/PHP70scalartypehints”;
analyzer[] = “Php/PHP71scalartypehints”;
analyzer[] = “Php/PHP72scalartypehints”;
analyzer[] = “Php/PHP73LastEmptyArgument”;
analyzer[] = “Php/ParenthesisAsParameter”;
analyzer[] = “Php/Php54RemovedFunctions”;
analyzer[] = “Php/Php55NewFunctions”;
analyzer[] = “Php/Php56NewFunctions”;
analyzer[] = “Php/Php70NewClasses”;
analyzer[] = “Php/Php70NewFunctions”;
analyzer[] = “Php/Php70NewInterfaces”;
analyzer[] = “Php/Php71NewClasses”;
analyzer[] = “Php/Php72NewClasses”;
analyzer[] = “Php/Php73NewFunctions”;
analyzer[] = “Php/Php7RelaxedKeyword”;
analyzer[] = “Php/StaticclassUsage”;
analyzer[] = “Php/TrailingComma”;
analyzer[] = “Php/TypedPropertyUsage”;
analyzer[] = “Php/UnicodeEscapePartial”;
analyzer[] = “Php/UnicodeEscapeSyntax”;
analyzer[] = “Php/UnpackingInsideArrays”;
analyzer[] = “Php/UseNullableType”;
analyzer[] = “Php/debugInfoUsage”;
analyzer[] = “Structures/BreakNonInteger”;
analyzer[] = “Structures/CalltimePassByReference”;
analyzer[] = “Structures/ConstantScalarExpression”;
analyzer[] = “Structures/ContinueIsForLoop”;
analyzer[] = “Structures/CryptWithoutSalt”;
analyzer[] = “Structures/DereferencingAS”;
analyzer[] = “Structures/ForeachWithList”;
analyzer[] = “Structures/IssetWithConstant”;
analyzer[] = “Structures/NoGetClassNull”;
analyzer[] = “Structures/PHP7Dirname”;
analyzer[] = “Structures/SwitchWithMultipleDefault”;
analyzer[] = “Structures/VariableGlobal”;
analyzer[] = “Type/MalformedOctal”;
analyzer[] = “Variables/Php5IndirectExpression”;
analyzer[] = “Variables/Php7IndirectExpression”;|
```

CompatibilityPHP55 This ruleset centralizes all analysis for the migration from PHP 5.4 to 5.5.

```
[CompatibilityPHP55]
    analyzer[] = “Classes/Anonymous”;
    analyzer[] = “Classes/CantInheritAbstractMethod”;
    analyzer[] = “Classes/ChildRemoveTypehint”;
    analyzer[] = “Classes/ConstVisibilityUsage”;
    analyzer[] = “Classes/IntegerAsProperty”;
    analyzer[] = “Classes/NonStaticMethodsCalledStatic”;
    analyzer[] = “Classes/NullOnNew”;
    analyzer[] = “Exceptions/MultipleCatch”;
    analyzer[] = “Extensions/Extapc”;
    analyzer[] = “Extensions/Extmysql”;
    analyzer[] = “Functions/GeneratorCannotReturn”;
    analyzer[] = “Functions/MultipleSameArguments”;
    analyzer[] = “Namespaces/UseFunctionsConstants”;
    analyzer[] = “Php/ClassConstWithArray”;
    analyzer[] = “Php/CoalesceEqual”;
    analyzer[] = “Php/ConcatAndAddition”;
    analyzer[] = “Php/ConstWithArray”;
    analyzer[] = “Php/DefineWithArray”;
    analyzer[] = “Php/DirectCallToClone”;
    analyzer[] = “Php/EllipsisUsage”;
    analyzer[] = “Php/ExponentUsage”;
    analyzer[] = “Php/FlexibleHeredoc”;
    analyzer[] = “Php/GroupUseDeclaration”;
    analyzer[] = “Php/GroupUseTrailingComma”;
    analyzer[] = “Php/HashAlgos53”;
    analyzer[] = “Php/HashAlgos54”;
    analyzer[] = “Php/HashAlgos71”;
    analyzer[] = “Php/ListShortSyntax”;
    analyzer[] = “Php/ListWithKeys”;
    analyzer[] = “Php/ListWithReference”;
    analyzer[] = “Php/NoListWithString”;
    analyzer[] = “Php/NoReferenceForStaticProperty”;
    analyzer[] = “Php/NoReturnForGenerator”;
    analyzer[] = “Php/NoStringWithAppend”;
    analyzer[] = “Php/NoSubstrMinusOne”;
    analyzer[] = “Php/PHP70scalartypehints”;
    analyzer[] = “Php/PHP71scalartypehints”;
    analyzer[] = “Php/PHP72scalartypehints”;
    analyzer[] = “Php/PHP73LastEmptyArgument”;
    analyzer[] = “Php/ParenthesisAsParameter”;
    analyzer[] = “Php/Password55”;
    analyzer[] = “Php/Php55RemovedFunctions”;
    analyzer[] = “Php/Php56NewFunctions”;
    analyzer[] = “Php/Php70NewClasses”;
    analyzer[] = “Php/Php70NewFunctions”;
    analyzer[] = “Php/Php70NewInterfaces”;
    analyzer[] = “Php/Php71NewClasses”;
    analyzer[] = “Php/Php72NewClasses”;
    analyzer[] = “Php/Php73NewFunctions”;
    analyzer[] = “Php/Php7RelaxedKeyword”;
    analyzer[] = “Php/TrailingComma”;
    analyzer[] = “Php/TypedPropertyUsage”;
    analyzer[] = “Php/UnicodeEscapePartial”;
    analyzer[] = “Php/UnicodeEscapeSyntax”;
    analyzer[] = “Php/UnpackingInsideArrays”;
    analyzer[] = “Php/UseNullableType”;
    analyzer[] = “Php/debugInfoUsage”;
    analyzer[] = “Structures/ConstantScalarExpression”;
    analyzer[] = “Structures/ContinueIsForLoop”;
    analyzer[] = “Structures/IssetWithConstant”;
    analyzer[] = “Structures/NoGetClassNull”;
    analyzer[] = “Structures/PHP7Dirname”;
    analyzer[] = “Structures/SwitchWithMultipleDefault”;
    analyzer[] = “Structures/VariableGlobal”;
    analyzer[] = “Type/MalformedOctal”;
    analyzer[] = “Variables/Php5IndirectExpression”;
    analyzer[] = “Variables/Php7IndirectExpression”;|
```

CompatibilityPHP56 This ruleset centralizes all analysis for the migration from PHP 5.5 to 5.6.

```
[CompatibilityPHP56]
    analyzer[] = “Classes/Anonymous”;
    analyzer[] = “Classes/CantInheritAbstractMethod”;
    analyzer[] = “Classes/ChildRemoveTypehint”;
    analyzer[] = “Classes/ConstVisibilityUsage”;
    analyzer[] = “Classes/IntegerAsProperty”;
    analyzer[] = “Classes/NonStaticMethodsCalledStatic”;
    analyzer[] = “Classes/NullOnNew”;
    analyzer[] = “Exceptions/MultipleCatch”;
    analyzer[] = “Functions/GeneratorCannotReturn”;
    analyzer[] = “Functions/MultipleSameArguments”;
    analyzer[] = “Php/CoalesceEqual”;
    analyzer[] = “Php/ConcatAndAddition”;
    analyzer[] = “Php/DefineWithArray”;
    analyzer[] = “Php/DirectCallToClone”;
    analyzer[] = “Php/FlexibleHeredoc”;
    analyzer[] = “Php/GroupUseDeclaration”;
    analyzer[] = “Php/GroupUseTrailingComma”;
    analyzer[] = “Php/HashAlgos53”;
    analyzer[] = “Php/HashAlgos54”;
    analyzer[] = “Php/HashAlgos71”;
    analyzer[] = “Php/ListShortSyntax”;
    analyzer[] = “Php/ListWithKeys”;
    analyzer[] = “Php/ListWithReference”;
    analyzer[] = “Php/NoListWithString”;
    analyzer[] = “Php/NoReferenceForStaticProperty”;
    analyzer[] = “Php/NoReturnForGenerator”;
    analyzer[] = “Php/NoStringWithAppend”;
    analyzer[] = “Php/NoSubstrMinusOne”;
    analyzer[] = “Php/PHP70scalartypehints”;
    analyzer[] = “Php/PHP71scalartypehints”;
    analyzer[] = “Php/PHP72scalartypehints”;
    analyzer[] = “Php/PHP73LastEmptyArgument”;
    analyzer[] = “Php/ParenthesisAsParameter”;
    analyzer[] = “Php/Php70NewClasses”;
    analyzer[] = “Php/Php70NewFunctions”;
    analyzer[] = “Php/Php70NewInterfaces”;
    analyzer[] = “Php/Php71NewClasses”;
    analyzer[] = “Php/Php72NewClasses”;
    analyzer[] = “Php/Php73NewFunctions”;
    analyzer[] = “Php/Php7RelaxedKeyword”;
    analyzer[] = “Php/Php80OnlyTypeHints”;
    analyzer[] = “Php/RawPostDataUsage”;
    analyzer[] = “Php/TrailingComma”;
    analyzer[] = “Php/TypedPropertyUsage”;
    analyzer[] = “Php/UnicodeEscapePartial”;
    analyzer[] = “Php/UnicodeEscapeSyntax”;
    analyzer[] = “Php/UnpackingInsideArrays”;
    analyzer[] = “Php/UseNullableType”;
    analyzer[] = “Structures/ContinueIsForLoop”;
    analyzer[] = “Structures/IssetWithConstant”;
    analyzer[] = “Structures/NoGetClassNull”;
    analyzer[] = “Structures/PHP7Dirname”;
    analyzer[] = “Structures/SwitchWithMultipleDefault”;
    analyzer[] = “Structures/VariableGlobal”;
    analyzer[] = “Type/MalformedOctal”;
    analyzer[] = “Variables/Php5IndirectExpression”;
    analyzer[] = “Variables/Php7IndirectExpression”;|
```

CompatibilityPHP70 This ruleset centralizes all analysis for the migration from PHP 5.6 to 7.0. __________________

```
[CompatibilityPHP70]
    analyzer[] = “Classes/CantInheritAbstractMethod”;
    analyzer[] = “Classes/ChildRemoveTypehint”;
    analyzer[] = “Classes/ConstVisibilityUsage”;
```

```
analyzer[] = “Classes/IntegerAsProperty”;
analyzer[] = “Classes/toStringPss”;
analyzer[] = “Exceptions/MultipleCatch”;
analyzer[] = “Extensions/Extereg”;
analyzer[] = “Functions/funcGetArgModified”;
analyzer[] = “Php/CoalesceEqual”;
analyzer[] = “Php/ConcatAndAddition”;
analyzer[] = “Php/EmptyList”;
analyzer[] = “Php/FlexibleHeredoc”;
analyzer[] = “Php/ForeachDontChangePointer”;
analyzer[] = “Php/GlobalWithoutSimpleVariable”;
analyzer[] = “Php/GroupUseTrailingComma”;
analyzer[] = “Php/HashAlgos53”;
analyzer[] = “Php/HashAlgos54”;
analyzer[] = “Php/HashAlgos71”;
analyzer[] = “Php/ListShortSyntax”;
analyzer[] = “Php/ListWithAppends”;
analyzer[] = “Php/ListWithKeys”;
analyzer[] = “Php/ListWithReference”;
analyzer[] = “Php/NoReferenceForStaticProperty”;
analyzer[] = “Php/NoSubstrMinusOne”;
analyzer[] = “Php/PHP71scalartypehints”;
analyzer[] = “Php/PHP72scalartypehints”;
analyzer[] = “Php/PHP73LastEmptyArgument”;
analyzer[] = “Php/Php70RemovedDirective”;
analyzer[] = “Php/Php70RemovedFunctions”;
analyzer[] = “Php/Php71NewClasses”;
analyzer[] = “Php/Php72NewClasses”;
analyzer[] = “Php/Php73NewFunctions”;
analyzer[] = “Php/Php80OnlyTypeHints”;
analyzer[] = “Php/Php80UnionTypehint”;
analyzer[] = “Php/ReservedKeywords7”;
analyzer[] = “Php/SetExceptionHandlerPHP7”;
analyzer[] = “Php/TrailingComma”;
analyzer[] = “Php/TypedPropertyUsage”;
analyzer[] = “Php/UnpackingInsideArrays”;
analyzer[] = “Php/UseNullableType”;
analyzer[] = “Php/UsortSorting”;
analyzer[] = “Structures/BreakOutsideLoop”;
analyzer[] = “Structures/ContinueIsForLoop”;
analyzer[] = “Structures/McryptcreateivWithoutOption”;
analyzer[] = “Structures/NoGetClassNull”;
analyzer[] = “Structures/SetlocaleNeedsConstants”;
analyzer[] = “Structures/pregOptionE”;
analyzer[] = “Type/HexadecimalString”;
analyzer[] = “Variables/Php7IndirectExpression”;|
```

CompatibilityPHP71 This ruleset centralizes all analysis for the migration from PHP 7.0 to 7.1.

```
[CompatibilityPHP71]
        analyzer[] = “Arrays/StringInitialization”;
        analyzer[] = “Classes/CantInheritAbstractMethod”;
        analyzer[] = “Classes/ChildRemoveTypehint”;
        analyzer[] = “Classes/IntegerAsProperty”;
        analyzer[] = “Classes/UsingThisOutsideAClass”;
        analyzer[] = “Extensions/Extmcrypt”;
        analyzer[] = “Php/BetterRand”;
        analyzer[] = “Php/CoalesceEqual”;
        analyzer[] = “Php/ConcatAndAddition”;
        analyzer[] = “Php/FlexibleHeredoc”;
        analyzer[] = “Php/GroupUseTrailingComma”;
        analyzer[] = “Php/HashAlgos53”;
        analyzer[] = “Php/HashAlgos54”;
        analyzer[] = “Php/ListWithReference”;
        analyzer[] = “Php/NoReferenceForStaticProperty”;
        analyzer[] = “Php/PHP72scalartypehints”;
        analyzer[] = “Php/PHP73LastEmptyArgument”;
        analyzer[] = “Php/Php70RemovedDirective”;
        analyzer[] = “Php/Php70RemovedFunctions”;
        analyzer[] = “Php/Php71NewFunctions”;
        analyzer[] = “Php/Php71RemovedDirective”;
        analyzer[] = “Php/Php71microseconds”;
        analyzer[] = “Php/Php72NewClasses”;
        analyzer[] = “Php/Php73NewFunctions”;
        analyzer[] = “Php/Php80OnlyTypeHints”;
        analyzer[] = “Php/Php80UnionTypehint”;
        analyzer[] = “Php/SignatureTrailingComma”;
        analyzer[] = “Php/TrailingComma”;
        analyzer[] = “Php/TypedPropertyUsage”;
        analyzer[] = “Php/UnpackingInsideArrays”;
        analyzer[] = “Structures/ContinueIsForLoop”;
        analyzer[] = “Structures/NoGetClassNull”;
        analyzer[] = “Structures/NoSubstrOne”;
        analyzer[] = “Structures/pregOptionE”;
        analyzer[] = “Type/HexadecimalString”;
        analyzer[] = “Type/OctalInString”;|
```

CompatibilityPHP72 This ruleset centralizes all analysis for the migration from PHP 7.1 to 7.2.

```
[CompatibilityPHP72]
        analyzer[] = “Constants/UndefinedConstants”;
        analyzer[] = “Php/AvoidSetErrorHandlerContextArg”;
```

```
    analyzer[] = “Php/CoalesceEqual”;
    analyzer[] = “Php/ConcatAndAddition”;
    analyzer[] = “Php/FlexibleHeredoc”;
    analyzer[] = “Php/HashAlgos53”;
    analyzer[] = “Php/HashAlgos54”;
    analyzer[] = “Php/HashUsesObjects”;
    analyzer[] = “Php/ListWithReference”;
    analyzer[] = “Php/NoReferenceForStaticProperty”;
    analyzer[] = “Php/PHP73LastEmptyArgument”;
    analyzer[] = “Php/Php72Deprecation”;
    analyzer[] = “Php/Php72NewClasses”;
    analyzer[] = “Php/Php72NewConstants”;
    analyzer[] = “Php/Php72NewFunctions”;
    analyzer[] = “Php/Php72ObjectKeyword”;
    analyzer[] = “Php/Php72RemovedFunctions”;
    analyzer[] = “Php/Php73NewFunctions”;
    analyzer[] = “Php/Php80OnlyTypeHints”;
    analyzer[] = “Php/Php80UnionTypehint”;
    analyzer[] = “Php/SignatureTrailingComma”;
    analyzer[] = “Php/ThrowWasAnExpression”;
    analyzer[] = “Php/TrailingComma”;
    analyzer[] = “Php/TypedPropertyUsage”;
    analyzer[] = “Php/UnpackingInsideArrays”;
    analyzer[] = “Structures/CanCountNonCountable”;
    analyzer[] = “Structures/ContinueIsForLoop”;
    analyzer[] = “Structures/NoGetClassNull”;
    analyzer[] = “Structures/pregOptionE”;|
```

CompatibilityPHP73 This ruleset centralizes all analysis for the migration from PHP 7.2 to 7.3.

```
[CompatibilityPHP73]
    analyzer[] = “Constants/CaseInsensitiveConstants”;
    analyzer[] = “Php/AssertFunctionIsReserved”;
    analyzer[] = “Php/CoalesceEqual”;
    analyzer[] = “Php/CompactInexistant”;
    analyzer[] = “Php/ConcatAndAddition”;
    analyzer[] = “Php/IntegerSeparatorUsage”;
    analyzer[] = “Php/Php73NewFunctions”;
    analyzer[] = “Php/Php73RemovedFunctions”;
    analyzer[] = “Php/Php74NewDirective”;
    analyzer[] = “Php/Php80OnlyTypeHints”;
    analyzer[] = “Php/Php80UnionTypehint”;
    analyzer[] = “Php/SignatureTrailingComma”;
    analyzer[] = “Php/ThrowWasAnExpression”;
    analyzer[] = “Php/TypedPropertyUsage”;
    analyzer[] = “Php/UnknownPcre2Option”;
    analyzer[] = “Php/UnpackingInsideArrays”;
    analyzer[] = “Structures/ContinueIsForLoop”;
    analyzer[] = “Structures/DontReadAndWriteInOneExpression”;|
```

CompatibilityPHP74 This ruleset centralizes all analysis for the migration from PHP 7.3 to 7.4.

```
[CompatibilityPHP74]
    analyzer[] = “Functions/UnbindingClosures”;
    analyzer[] = “Php/ArrayKeyExistsWithObjects”;
    analyzer[] = “Php/ConcatAndAddition”;
    analyzer[] = “Php/DetectCurrentClass”;
    analyzer[] = “Php/FilterToAddSlashes”;
    analyzer[] = “Php/HashAlgos74”;
    analyzer[] = “Php/IdnUts46”;
    analyzer[] = “Php/NestedTernaryWithoutParenthesis”;
    analyzer[] = “Php/NoMoreCurlyArrays”;
    analyzer[] = “Php/Php74Deprecation”;
    analyzer[] = “Php/Php74NewClasses”;
    analyzer[] = “Php/Php74NewConstants”;
    analyzer[] = “Php/Php74NewFunctions”;
    analyzer[] = “Php/Php74RemovedDirective”;
    analyzer[] = “Php/Php74RemovedFunctions”;
    analyzer[] = “Php/Php74ReservedKeyword”;
    analyzer[] = “Php/Php74mbstrrpos3rdArg”;
    analyzer[] = “Php/Php80NewFunctions”;
    analyzer[] = “Php/Php80OnlyTypeHints”;
    analyzer[] = “Php/Php80UnionTypehint”;
    analyzer[] = “Php/Php80VariableSyntax”;
    analyzer[] = “Php/ReflectionExportIsDeprecated”;
    analyzer[] = “Php/ScalarAreNotArrays”;
    analyzer[] = “Php/SignatureTrailingComma”;
    analyzer[] = “Php/ThrowWasAnExpression”;
    analyzer[] = “Php/UseMatch”;
    analyzer[] = “Structures/CurlVersionNow”;
    analyzer[] = “Structures/DontReadAndWriteInOneExpression”;
    analyzer[] = “Structures/OpensslRandomPseudoByteSecondArg”;|
```

CompatibilityPHP80 This ruleset centralizes all analysis for the migration from PHP 7.4 to 8.0.

```
[CompatibilityPHP80]
    analyzer[] = “Arrays/NegativeStart”;
    analyzer[] = “Classes/OldStyleConstructor”;
    analyzer[] = “Functions/MismatchParameterName”;
    analyzer[] = “Functions/NullableWithConstant”;
```

```
        analyzer[] = “Php/CastUnsetUsage”;
        analyzer[] = “Php/ConcatAndAddition”;
        analyzer[] = “Php/Php80RemovedConstant”;
        analyzer[] = “Php/Php80RemovedDirective”;
        analyzer[] = “Php/Php80RemovedFunctions”;
        analyzer[] = “Php/PhpErrorMsgUsage”;
        analyzer[] = “Structures/UnsupportedTypesWithOperators”;|
```

Dead code This ruleset focuses on dead code : expressions or even structures that are written, valid but never used.

```
[Dead code]
        analyzer[] = “Classes/CantExtendFinal”;
        analyzer[] = “Classes/LocallyUnusedProperty”;
        analyzer[] = “Classes/UnresolvedCatch”;
        analyzer[] = “Classes/UnresolvedInstanceof”;
        analyzer[] = “Classes/UnusedClass”;
        analyzer[] = “Classes/UnusedMethods”;
        analyzer[] = “Classes/UnusedPrivateMethod”;
        analyzer[] = “Classes/UnusedPrivateProperty”;
        analyzer[] = “Classes/UnusedProtectedMethods”;
        analyzer[] = “Constants/UnusedConstants”;
        analyzer[] = “Exceptions/AlreadyCaught”;
        analyzer[] = “Exceptions/CaughtButNotThrown”;
        analyzer[] = “Exceptions/Rethrown”;
        analyzer[] = “Exceptions/Unthrown”;
        analyzer[] = “Functions/UnusedFunctions”;
        analyzer[] = “Functions/UnusedInheritedVariable”;
        analyzer[] = “Functions/UnusedReturnedValue”;
        analyzer[] = “Functions/UselessTypeCheck”;
        analyzer[] = “Interfaces/UnusedInterfaces”;
        analyzer[] = “Namespaces/EmptyNamespace”;
        analyzer[] = “Namespaces/UnusedUse”;
        analyzer[] = “Structures/EmptyLines”;
        analyzer[] = “Structures/UnreachableCode”;
        analyzer[] = “Structures/UnsetInForeach”;
        analyzer[] = “Structures/UnusedLabel”;
        analyzer[] = “Traits/SelfUsingTrait”;|
```

LintButWontExec This ruleset focuses on PHP code that lint (php -l), but that will not run. As such, this ruleset tries to go further than PHP, by connecting files, just like during execution.

```
[LintButWontExec]
        analyzer[] = “Classes/AbstractOrImplements”;
        analyzer[] = “Classes/CloneWithNonObject”;
```

```
    analyzer[] = “Classes/CouldBeStringable”;
    analyzer[] = “Classes/Finalclass”;
    analyzer[] = “Classes/Finalmethod”;
    analyzer[] = “Classes/IncompatibleSignature”;
    analyzer[] = “Classes/MethodSignatureMustBeCompatible”;
    analyzer[] = “Classes/MismatchProperties”;
    analyzer[] = “Classes/MutualExtension”;
    analyzer[] = “Classes/NoMagicWithArray”;
    analyzer[] = “Classes/NoPSSOutsideClass”;
    analyzer[] = “Classes/NoSelfReferencingConstant”;
    analyzer[] = “Classes/RaisedAccessLevel”;
    analyzer[] = “Classes/UsingThisOutsideAClass”;
    analyzer[] = “Classes/WrongTypedPropertyInit”;
    analyzer[] = “Exceptions/CantThrow”;
    analyzer[] = “Functions/MismatchTypeAndDefault”;
    analyzer[] = “Functions/MustReturn”;
    analyzer[] = “Functions/OnlyVariableForReference”;
    analyzer[] = “Functions/TypehintMustBeReturned”;
    analyzer[] = “Interfaces/CantImplementTraversable”;
    analyzer[] = “Interfaces/ConcreteVisibility”;
    analyzer[] = “Interfaces/IsNotImplemented”;
    analyzer[] = “Interfaces/RepeatedInterface”;
    analyzer[] = “Traits/MethodCollisionTraits”;
    analyzer[] = “Traits/TraitNotFound”;
    analyzer[] = “Traits/UndefinedInsteadof”;
    analyzer[] = “Traits/UndefinedTrait”;
    analyzer[] = “Traits/UselessAlias”;|
```

Performances This ruleset focuses on performances issues : anything that slows the code’s execution.

```
[Performances]
    analyzer[] = “Arrays/GettingLastElement”;
    analyzer[] = “Arrays/SliceFirst”;
    analyzer[] = “Classes/MakeMagicConcrete”;
    analyzer[] = “Classes/UseClassOperator”;
    analyzer[] = “Functions/Closure2String”;
    analyzer[] = “Performances/ArrayKeyExistsSpeedup”;
    analyzer[] = “Performances/ArrayMergeInLoops”;
    analyzer[] = “Performances/Autoappend”;
    analyzer[] = “Performances/AvoidArrayPush”;
    analyzer[] = “Performances/CacheVariableOutsideLoop”;
    analyzer[] = “Performances/CsvInLoops”;
    analyzer[] = “Performances/DoInBase”;
    analyzer[] = “Performances/DoubleArrayFlip”;
    analyzer[] = “Performances/FetchOneRowFormat”;
    analyzer[] = “Performances/IssetWholeArray”;
    analyzer[] = “Performances/JoinFile”;
    analyzer[] = “Performances/MakeOneCall”;
    analyzer[] = “Performances/MbStringInLoop”;
    analyzer[] = “Performances/NoConcatInLoop”;
    analyzer[] = “Performances/NoGlob”;
    analyzer[] = “Performances/NotCountNull”;
    analyzer[] = “Performances/OptimizeExplode”;
    analyzer[] = “Performances/PHP7EncapsedStrings”;
    analyzer[] = “Performances/Php74ArrayKeyExists”;
    analyzer[] = “Performances/PrePostIncrement”;
    analyzer[] = “Performances/RegexOnArrays”;
    analyzer[] = “Performances/RegexOnCollector”;
    analyzer[] = “Performances/SimpleSwitch”;
    analyzer[] = “Performances/SlowFunctions”;
    analyzer[] = “Performances/SubstrFirst”;
    analyzer[] = “Performances/UseBlindVar”;
    analyzer[] = “Performances/timeVsstrtotime”;
    analyzer[] = “Php/ShouldUseArrayColumn”;
    analyzer[] = “Php/ShouldUseFunction”;
    analyzer[] = “Php/UsePathinfoArgs”;
    analyzer[] = “Structures/CouldUseShortAssignation”;
    analyzer[] = “Structures/EchoWithConcat”;
    analyzer[] = “Structures/EvalUsage”;
    analyzer[] = “Structures/ForWithFunctioncall”;
    analyzer[] = “Structures/GlobalOutsideLoop”;
    analyzer[] = “Structures/NoArrayUnique”;
    analyzer[] = “Structures/NoAssignationInFunction”;
    analyzer[] = “Structures/NoSubstrOne”;
    analyzer[] = “Structures/Noscream”;
    analyzer[] = “Structures/SimplePreg”;
    analyzer[] = “Structures/WhileListEach”;|
```

Rector RectorPHP is a reconstructor tool. It applies modifications in the PHP code automatically. Exakat finds results which may be automatically updated with rector. ______

```
[Rector]
    analyzer[] = “Php/IsAWithString”;
    analyzer[] = “Structures/ElseIfElseif”;
    analyzer[] = “Structures/ShouldPreprocess”;|
```

Security This ruleset focuses on code security. ________

```
[Security]
    analyzer[] = “Functions/HardcodedPasswords”;
    analyzer[] = “Php/BetterRand”;
```

```
analyzer[] = “Security/AnchorRegex”;
analyzer[] = “Security/AvoidThoseCrypto”;
analyzer[] = “Security/CompareHash”;
analyzer[] = “Security/ConfigureExtract”;
analyzer[] = “Security/CryptoKeyLength”;
analyzer[] = “Security/CurlOptions”;
analyzer[] = “Security/DirectInjection”;
analyzer[] = “Security/DontEchoError”;
analyzer[] = “Security/DynamicDl”;
analyzer[] = “Security/EncodedLetters”;
analyzer[] = “Security/FilterInputSource”;
analyzer[] = “Security/IndirectInjection”;
analyzer[] = “Security/IntegerConversion”;
analyzer[] = “Security/KeepFilesRestricted”;
analyzer[] = “Security/MinusOneOnError”;
analyzer[] = “Security/MkdirDefault”;
analyzer[] = “Security/MoveUploadedFile”;
analyzer[] = “Security/NoEntIgnore”;
analyzer[] = “Security/NoNetForXmlLoad”;
analyzer[] = “Security/NoSleep”;
analyzer[] = “Security/NoWeakSSLCrypto”;
analyzer[] = “Security/RegisterGlobals”;
analyzer[] = “Security/SafeHttpHeaders”;
analyzer[] = “Security/SessionLazyWrite”;
analyzer[] = “Security/SetCookieArgs”;
analyzer[] = “Security/ShouldUsePreparedStatement”;
analyzer[] = “Security/ShouldUseSessionRegenerateId”;
analyzer[] = “Security/Sqlite3RequiresSingleQuotes”;
analyzer[] = “Security/UnserializeSecondArg”;
analyzer[] = “Security/UploadFilenameInjection”;
analyzer[] = “Security/parseUrlWithoutParameters”;
analyzer[] = “Structures/EvalUsage”;
analyzer[] = “Structures/EvalWithoutTry”;
analyzer[] = “Structures/Fallthrough”;
analyzer[] = “Structures/NoHardcodedHash”;
analyzer[] = “Structures/NoHardcodedIp”;
analyzer[] = “Structures/NoHardcodedPort”;
analyzer[] = “Structures/NoReturnInFinally”;
analyzer[] = “Structures/PhpinfoUsage”;
analyzer[] = “Structures/RandomWithoutTry”;
analyzer[] = “Structures/VardumpUsage”;
analyzer[] = “Structures/pregOptionE”;|
```

Semantics This ruleset focuses on human interpretation of the code. It reviews special values of literals, and named structures. _________

```
[Semantics]
        analyzer[] = “Arrays/WeirdIndex”;
        analyzer[] = “Functions/FnArgumentVariableConfusion”;
        analyzer[] = “Functions/MismatchParameterAndType”;
        analyzer[] = “Functions/OneLetterFunctions”;
        analyzer[] = “Functions/ParameterHiding”;
        analyzer[] = “Functions/PrefixToType”;
        analyzer[] = “Functions/SemanticTyping”;
        analyzer[] = “Functions/WrongTypehintedName”;
        analyzer[] = “Php/ClassFunctionConfusion”;
        analyzer[] = “Structures/PropertyVariableConfusion”;
        analyzer[] = “Type/DuplicateLiteral”;
        analyzer[] = “Type/SimilarIntegers”;
        analyzer[] = “Variables/VariableOneLetter”;|
```

Suggestions This ruleset focuses on possibly better syntax than the one currently used. Those may be code modernization, alternatives, more efficient solutions, or simply left over from older versions. ___________

```
[Suggestions]
        analyzer[] = “Arrays/RandomlySortedLiterals”;
        analyzer[] = “Arrays/ShouldPreprocess”;
        analyzer[] = “Arrays/SliceFirst”;
        analyzer[] = “Classes/CancelCommonMethod”;
        analyzer[] = “Classes/ParentFirst”;
        analyzer[] = “Classes/ShouldDeepClone”;
        analyzer[] = “Classes/ShouldHaveDestructor”;
        analyzer[] = “Classes/ShouldUseSelf”;
        analyzer[] = “Classes/TooManyChildren”;
        analyzer[] = “Classes/UnitializedProperties”;
        analyzer[] = “Classes/UselessTypehint”;
        analyzer[] = “Constants/CouldBeConstant”;
        analyzer[] = “Exceptions/CouldUseTry”;
        analyzer[] = “Exceptions/LargeTryBlock”;
        analyzer[] = “Exceptions/OverwriteException”;
        analyzer[] = “Functions/AddDefaultValue”;
        analyzer[] = “Functions/Closure2String”;
        analyzer[] = “Functions/CouldBeStaticClosure”;
        analyzer[] = “Functions/CouldCentralize”;
        analyzer[] = “Functions/NeverUsedParameter”;
        analyzer[] = “Functions/NoReturnUsed”;
        analyzer[] = “Functions/TooManyParameters”;
        analyzer[] = “Functions/TooMuchIndented”;
        analyzer[] = “Functions/UselessDefault”;
        analyzer[] = “Interfaces/AlreadyParentsInterface”;
        analyzer[] = “Interfaces/UnusedInterfaces”;
        analyzer[] = “Namespaces/AliasConfusion”;
```

```
analyzer[] = “Namespaces/CouldUseAlias”;
analyzer[] = “Patterns/AbstractAway”;
analyzer[] = “Performances/ArrayKeyExistsSpeedup”;
analyzer[] = “Performances/IssetWholeArray”;
analyzer[] = “Performances/SubstrFirst”;
analyzer[] = “Php/AvoidReal”;
analyzer[] = “Php/CompactInexistant”;
analyzer[] = “Php/CouldUseIsCountable”;
analyzer[] = “Php/CouldUsePromotedProperties”;
analyzer[] = “Php/DetectCurrentClass”;
analyzer[] = “Php/ImplodeOneArg”;
analyzer[] = “Php/IssetMultipleArgs”;
analyzer[] = “Php/LogicalInLetters”;
analyzer[] = “Php/NewExponent”;
analyzer[] = “Php/PregMatchAllFlag”;
analyzer[] = “Php/ReturnWithParenthesis”;
analyzer[] = “Php/ShouldPreprocess”;
analyzer[] = “Php/ShouldUseArrayColumn”;
analyzer[] = “Php/ShouldUseArrayFilter”;
analyzer[] = “Php/ShouldUseCoalesce”;
analyzer[] = “Php/UseDateTimeImmutable”;
analyzer[] = “Php/UseSessionStartOptions”;
analyzer[] = “Structures/BasenameSuffix”;
analyzer[] = “Structures/BooleanStrictComparison”;
analyzer[] = “Structures/CouldUseArrayFillKeys”;
analyzer[] = “Structures/CouldUseArrayUnique”;
analyzer[] = “Structures/CouldUseCompact”;
analyzer[] = “Structures/CouldUseDir”;
analyzer[] = “Structures/DirectlyUseFile”;
analyzer[] = “Structures/DontCompareTypedBoolean”;
analyzer[] = “Structures/DontLoopOnYield”;
analyzer[] = “Structures/DropElseAfterReturn”;
analyzer[] = “Structures/EchoWithConcat”;
analyzer[] = “Structures/EmptyWithExpression”;
analyzer[] = “Structures/FunctionPreSubscripting”;
analyzer[] = “Structures/JsonWithOption”;
analyzer[] = “Structures/ListOmissions”;
analyzer[] = “Structures/LongBlock”;
analyzer[] = “Structures/MismatchedTernary”;
analyzer[] = “Structures/MultipleUnset”;
analyzer[] = “Structures/NamedRegex”;
analyzer[] = “Structures/NoNeedGetClass”;
analyzer[] = “Structures/NoParenthesisForLanguageConstruct”;
analyzer[] = “Structures/NoSubstrOne”;
analyzer[] = “Structures/OneIfIsSufficient”;
analyzer[] = “Structures/PHP7Dirname”;
analyzer[] = “Structures/PossibleIncrement”;
```

```
    analyzer[] = “Structures/RepeatedPrint”;
    analyzer[] = “Structures/ReuseVariable”;
    analyzer[] = “Structures/SGVariablesConfusion”;
    analyzer[] = “Structures/SetAside”;
    analyzer[] = “Structures/ShouldUseForeach”;
    analyzer[] = “Structures/ShouldUseMath”;
    analyzer[] = “Structures/ShouldUseOperator”;
    analyzer[] = “Structures/SubstrLastArg”;
    analyzer[] = “Structures/SubstrToTrim”;
    analyzer[] = “Structures/UnreachableCode”;
    analyzer[] = “Structures/UseArrayFunctions”;
    analyzer[] = “Structures/UseCaseValue”;
    analyzer[] = “Structures/UseCountRecursive”;
    analyzer[] = “Structures/UseListWithForeach”;
    analyzer[] = “Structures/UseUrlQueryFunctions”;
    analyzer[] = “Structures/WhileListEach”;
    analyzer[] = “Traits/MultipleUsage”;
    analyzer[] = “Variables/ComplexDynamicNames”;|
```

Top10 This ruleset is a selection of analysis, with the top 10 most common. Actually, it is a little larger than that.

```
[Top10]
    analyzer[] = “Classes/DontUnsetProperties”;
    analyzer[] = “Classes/UnitializedProperties”;
    analyzer[] = “Classes/UnresolvedInstanceof”;
    analyzer[] = “Constants/ConstRecommended”;
    analyzer[] = “Functions/ShouldYieldWithKey”;
    analyzer[] = “Performances/ArrayMergeInLoops”;
    analyzer[] = “Performances/CsvInLoops”;
    analyzer[] = “Performances/NoConcatInLoop”;
    analyzer[] = “Performances/SubstrFirst”;
    analyzer[] = “Php/AvoidReal”;
    analyzer[] = “Php/ConcatAndAddition”;
    analyzer[] = “Php/LetterCharsLogicalFavorite”;
    analyzer[] = “Php/LogicalInLetters”;
    analyzer[] = “Php/MissingSubpattern”;
    analyzer[] = “Structures/CouldUseStrrepeat”;
    analyzer[] = “Structures/DanglingArrayReferences”;
    analyzer[] = “Structures/FailingSubstrComparison”;
    analyzer[] = “Structures/ForWithFunctioncall”;
    analyzer[] = “Structures/NextMonthTrap”;
    analyzer[] = “Structures/NoChoice”;
    analyzer[] = “Structures/NoSubstrOne”;
    analyzer[] = “Structures/ObjectReferences”;
    analyzer[] = “Structures/QueriesInLoop”;
```

```
        analyzer[] = “Structures/RepeatedPrint”;
        analyzer[] = “Structures/StrposCompare”;
        analyzer[] = “Structures/UseListWithForeach”;
        analyzer[] = “Type/NoRealComparison”;
        analyzer[] = “Variables/VariableUsedOnce”;|
```

Typechecks This ruleset focuses on typehinting. Missing typehint, or inconsistent typehint, are reported. __________

```
[Typechecks]
        analyzer[] = “Classes/ChildRemoveTypehint”;
        analyzer[] = “Classes/FossilizedMethod”;
        analyzer[] = “Functions/BadTypehintRelay”;
        analyzer[] = “Functions/InsufficientTypehint”;
        analyzer[] = “Functions/MismatchTypeAndDefault”;
        analyzer[] = “Functions/MismatchedDefaultArguments”;
        analyzer[] = “Functions/MismatchedTypehint”;
        analyzer[] = “Functions/MissingTypehint”;
        analyzer[] = “Functions/NoClassAsTypehint”;
        analyzer[] = “Functions/ShouldBeTypehinted”;
        analyzer[] = “Functions/WrongArgumentType”;
        analyzer[] = “Functions/WrongTypeWithCall”;
        analyzer[] = “Interfaces/UselessInterfaces”;
        analyzer[] = “Php/NotScalarType”;
        analyzer[] = “Typehints/CouldBeCallable”;
        analyzer[] = “Typehints/CouldBeFloat”;
        analyzer[] = “Typehints/CouldBeInt”;
        analyzer[] = “Typehints/CouldBeIterable”;
        analyzer[] = “Typehints/CouldBeNull”;
        analyzer[] = “Typehints/CouldBeParent”;
        analyzer[] = “Typehints/CouldBeSelf”;
        analyzer[] = “Typehints/CouldBeString”;
        analyzer[] = “Typehints/CouldBeVoid”;|
```

php-cs-fixable [PHP-CS-fixer](https://github.com/FriendsOfPHP/PHP-CS-Fixer) is a tool to automatically fix PHP Coding Standards issues. It applies modifications in the PHP code automatically. Exakat finds results which may be automatically updated with php-cs-fixer. ______________

```
[php-cs-fixable]
        analyzer[] = “Classes/DontUnsetProperties”;
        analyzer[] = “Php/ImplodeOneArg”;
        analyzer[] = “Php/IsnullVsEqualNull”;
        analyzer[] = “Php/IssetMultipleArgs”;
        analyzer[] = “Php/LogicalInLetters”;
        analyzer[] = “Php/NewExponent”;
        analyzer[] = “Structures/CouldUseDir”;
```

analyzer[] = “Structures/ElseIfElseif”;
analyzer[] = “Structures/MultipleUnset”;
analyzer[] = “Structures/PHP7Dirname”;
analyzer[] = “Structures/UseConstant”;|