Document number: P3886R0
Date: 2025-10-21
Project: Programming Language C++
Audience: EWG
Reply-to: Michael Florian Hava[1] <mfh.cpp@gmail.com>

Wording for AT1-057

Abstract
This paper proposes a way to check for the implementation-defined replaceability of the contract-violation handler.

Tony Table

Before	Proposed

`#if __cpp_contracts`
`  #include <contracts>`
 	
`  void handle_contract_violation(`
`  const std::contracts::contract_violation &`
`  ); // ‼️ IFNDR if not replaceable`
 	
`#endif`
	`#if __cpp_contracts`
	`  #include <contracts>`
	
	`  #if __cpp_lib_replaceable_contract_violation_handler`
	
	`  void handle_contract_violation(`
	`  const std::contracts::contract_violation &`
	`  ); // ✅ guaranteed to be well-formed`
	
	`  #endif`
	`#endif`

Revisions
R0: Initial version

Motivation
Quoting N5014 **[basic.contract.handler]/3**:

> It is implementation-defined whether the contract-violation handler is replaceable (9.6.5). If the contract-violation handler is not replaceable, a declaration of a replacement function for the contract-violation handler is ill-formed, no diagnostic required.

Currently there is no in-code way to check whether the implementation supports replacing the contract-violation handler. This is a problem for portable codebases that may encounter either decision and want to provide a custom handler if supported.

Design Space
As the mere presence of a declaration can trigger IFNDR, the detection mechanism for whether the contract-violation handler is replaceable must be a feature test macro. The standard has two sets of feature test macros:

- Language features are predefined by the implementation (**[cpp.predefined]**).
- Library features are provided by the header `<version>` as well as the respective features header (**[version.syn]**).

[1] RISC Software GmbH, Softwarepark 32a, 4232 Hagenberg, Austria, michael.hava@risc-software.at

Design-wise the contract-violation handler is part of the language, yet realistically it will be provided by the standard library. Therefore this new feature test macro should be part of the latter group - mirroring the design of `__cpp_lib_freestanding_operator_new`, which also tracks library-aspects of a language feature.

Similarly to `__cpp_lib_freestanding_operator_new`, the newly proposed macro - we propose the name `__cpp_lib_replaceable_contract_violation_handler` – shall be 0 if the contract-violation handler is not replaceable.

Impact on the Standard

This is a pure extension that introduces one additional feature test macro.

Proposed Wording

Wording is relative to [N5014]. Additions are presented like this, removals like ~~this~~ and drafting notes like **this**.

[version.syn]

??.?.? Header <version> synopsis **[version.syn]**

2 **[DRAFTING NOTE: Add the following macro to the existing list.]**

```
#define __cpp_lib_replaceable_contract_violation_handler see below // freestanding, also in <contracts>
```

...

4 The macro `__cpp_lib_freestanding_operator_new` is defined to the integer literal `202306L` if all the default versions of the replaceable global allocation functions meet the requirements of a hosted implementation, and to the integer literal `0` otherwise (**[new.delete]**).

5 The macro `__cpp_lib_replaceable_contract_violation_handler` is defined to the integer literal YYYYMML if the contract violation handler is replaceable, and to the integer literal `0` otherwise.

[DRAFTING NOTE: Adjust the placeholder value as needed to denote the proposal's date of adoption.]

~~5~~6 *Recommended practice*: Freestanding implementations should only define a macro from <version> if the implementation provides the corresponding facility in its entirety.

Acknowledgements

Thanks to RISC Software GmbH for supporting this work.