

# Bloomberg Contract Roadmap

Joshua Berne - `jberne4@bloomberg.net`

2020-10-22

## Copyright Notice

©2020 Bloomberg L.P. Permission is granted to copy, distribute, and display this material, and to make derivative works and commercial use of it. The information in this material is provided “AS IS”, without warranty of any kind. Neither Bloomberg nor any employee guarantees the correctness or completeness of such information. Bloomberg, its employees, and its affiliated entities and persons shall not be liable, directly or indirectly, in any way, for any inaccuracies, errors or omissions in such information. Nothing herein should be interpreted as stating the opinions, policies, recommendations, or positions of Bloomberg.

## Why standardize contracts?

- Make the world a better place
- Tooling support - having third party static analysis tools able to verify contract checks for us
- Consistent and controllable third-party library behaviors
- Compiler support for optimizations based on contracts, and potentially to AVOID optimizations interfering with contracts.

## Current Phase - MVP

- Ability to declare `[[pre]]`, `[[post]]`, `[[assert]]` annotations
- Central violation handler invoked when “on” and violated
- No other semantics

```
int foo(int x)
  [[ pre : x >= 0 ]]
  [[ post r : r >= 0 ]]
{
  [[ assert : x * x > x ]];
  return x;
}
```

## Current Phase - MVP

```
[[ assert : x > 0 ]];  
  
if (x > 0) {  
    invoke_violation_handler();  
}
```

- Partial solution
- No “levels” - contracts on might break complexity guarantees, or those checks have to not be written
- No differentiating between “new” and “preexisting” contracts
- No extensibility
- No language-based ability to take more advantage of contracts
- Some usefulness can be embedded in a smart enough violation handler, but many of our use cases will not yet be met

## Fallen Heroes - C++20 Contracts

- P0542 - contracts that were merged - MVP + audit, axiom
- P1332 - introduced semantics
- P1429 - full semantic-supporting proposal on top of P0542
- P1607 - minimized, JUST semantics, accepted by EWG

## Fallen Heroes - C++20 Contracts

- P0542 - contracts that were merged - MVP + audit, axiom
- P1332 - introduced semantics
- P1429 - full semantic-supporting proposal on top of P0542
- P1607 - minimized, JUST semantics, accepted by EWG
- ignore - An “OFF” contract that has no impact on execution
- observe - An “ON” contract that lets the violation handler observe a violation.
- enforce - An “ON” contract that aborts if the violation handler returns.
- assume - A contract where violation is hard UB

## What is needed

- SG21 has 196 use cases for contracts defined (P1995)
- A plan to enabled satisfying most/all of them needs to be developed
- Our desire to migrate BSLS\_ASSERT and BSLS\_REVIEW to language-based contracts should be met (it is a subset of the SG21 use cases)
- Contract semantics should be selectable based on a combination of locally specified meta-attributes about a contract and build-time specified choices made by the person assembling an application.



## Next Phase - Levels, Roles, Labels?

```
BSLS_ASSERT_SAFE(X);  
[[ assert audit : X ]];  
  
BSLS_REVIEW(X)  
[[ assert review : X ]];  
  
BSLS_REVIEW_SAFE(X)  
[[ assert audit review : X ]];
```

- We need to independently control the semantics of each of these
- review's logic could be captured in code
- lots of other things we have identified uses for labelling contracts

## Next Phase - Plan

- Working with lock3 we have a prototype that supports everything in P0542+P1429+P1607
- That work can be leveraged with BSLS\_ASSERT and BSLS\_REVIEW today
- We are developing a similar proposal and prototype that subsumes P1429/P1607 and meets more use cases from P1995. That will hopefully begin review rounds and prototype implementation internally and with SG21 members over the next few months.