Introduction

- There are plenty of advantages using constants over the use of harcoded literals.
- Literals are currently used in a lot of places where constants should be defined instead.
- Defining new constants for those literals has an high impact (tens of thousands literals)

  ⇨ Tool needed for safety.

Comparison (1/5)

Literals

- Doesn't highlight the value's meaning.
- Spread everywhere into the source code.
- Analysis of the algorithm needed to validate their meaning and value afterward.

Constants

- The name highlights the value's meaning.
- Defined only in one place
- Much easier to validate their meaning and value.

Comparison (2/5)

Literals

- Changes are risky:
  - 💣 Risk to forget some instances of the literal
  - 💣 Risk to change some literals related to something else.

Constants

- Changes are safer:
  - ✓ Only needed once.

Comparison (3/5)

Literals

- Hard to keep control of sets of literals.

Constants

- Easier to maintain sets of literals:
  - ✓ Easier to check if unique Ids are… unique.
  - ✓ New unique Ids easier to define

Comparison (4/5)

Literals

- Hard to highlight the relationship among several literals

Constants

- Easy to highlight the relationship among constants, using one constant to define another one.

Comparison (5/5)

Literals

- No scope attached to a literal
- Dirty API prone
- Documentation holes prone

Constants

- Scope attached to the constant by both its name and its place of definition
- Helps to make an API public.
- Helps the documentation to refer to them

Safe strategy

- Comparison also highlights the need of a strong methodology for a safe move from literals to constants.
- There is a need for a tool preventing the developer to make mistakes for such repetitive tasks.

Tool's features (1/2)

- No need to type any value
  - ⇨ no typing mistake
- Constant choice help:
  - Builds list of existing constants
  - Able to compute complex expression in existing constant definitions
  - Detects both existing macros and constants
  - Supports prepocessing (#include, etc…)

Tool's features (2/2)

- List of literals sortable by differents criteria
  - ⇨Easy to gather related literals
- Minimum disturbance of other developer thank to a two steps strategy that supports a quick unlocking of files, if needed.
- Keeps track of the task's progress.

Strategy

- To avoid double constant definitions:
  - Work is done module by module, one person by module
  - Code reviews are done by people having done or having to do other modules
- Three code reviews during the course of the literal clean up.

Conclusion

Risks minimized thank to:

- A tool that checks the validity of the developer's actions
- A strong strategy with three code reviews
- Minimal disturbance of other developers thank to a two steps strategy.