Document Number: P0972R0
Date: 2018-02-28
Project: Programming Language C++
Reply-to: Billy Robert O'Neal III <bion@microsoft.com>

Audience: LEWG -> LWG

<chrono> zero(), min(), and max() should be noexcept

Motivation and scope:

Some Visual C++ standard library maintainers were surprised when we were unable to make directory_entry's default constructor noexcept, because chrono::time_point was not noexcept.

The standard appears to effectively require that duration / time_point / etc. have noexcept zero() / min() / max(), but proving this requires a winding path through requirements. Even a hypothetical future where <chrono> works with bignum types, such types should still be able to provide noexcept zero(), min(), and max().

Standard references herein are relative to N4741.

[time.traits.duration_values] says that duration_values<Rep> has static constexpr functions zero(), min(), and max(). Marking these constexpr with no parameters means any exceptions emitted would be highly surprising. These functions are wide contract, as they have no preconditions and no parameters.

duration and time_point have functions zero(), min(), and max() which effectively return duration_values<Rep> zero(), min(), and max() constructed inside a duration or time_point.

[time.duration]/4 prohibits any existing implementation from adding exceptions in their duration::zero() / min() / max(). [time.point.special] extends this to time_point::min() and max(). As a result, no existing implementation that throws in these locations should be possible. Therefore, we don't need any backwards compatibility notes or entries in Annex D.

Proposed wording:

Edit the signatures of [time.traits.duration_values] as follows:

```
template<class Rep>
    struct duration_values {
    public:
    static constexpr Rep zero() noexcept;
    static constexpr Rep min() noexcept;
    static constexpr Rep max() noexcept;
};

[...]

static constexpr Rep zero() noexcept;

[...]

static constexpr Rep min() noexcept;

[...]
```

```
static constexpr Rep max() noexcept;
```

Edit the signatures of [time.duration] as follows:

```
namespace std::chrono {
    template<class Rep, class Period = ratio<1>>
        class duration {

        [...]

        // 23.17.5.4, special values
        static constexpr duration zero() noexcept;
        static constexpr duration min() noexcept;
        static constexpr duration max() noexcept;
    };
}
```

Edit [time.duration.special] as follows:

```
static constexpr duration zero() noexcept;
```

-1- *Returns*: `duration(duration_values<rep>::zero())`.

```
static constexpr duration min() noexcept;
```

-2- *Returns*: `duration(duration_values<rep>::min())`.

```
static constexpr duration max() noexcept;
```

-3- *Returns*: `duration(duration_values<rep>::max())`.

Edit the synopsis of [time.point] as follows:

```
namespace std::chrono {
    template<class Clock, class Duration = typename Clock::duration>
    class time_point {

    [...]

    // 23.17.6.4, special values
    static constexpr time_point min() noexcept;
    static constexpr time_point max() noexcept;
    };
}
```

Edit [time.point.special] as follows:

```
static constexpr time_point min() noexcept;
```

-1- *Returns*: `time_point(duration::min())`.

```
static constexpr time_point max() noexcept;
```

-2- *Returns*: `time_point(duration::max())`.