

# Guava Release 23.0: Release Notes

- 23.0 was released on August 4, 2017.
- 23.0-rc1 was released on July 25, 2017.

(See [\[\[ReleaseHistory\]\]](#).)

## API documentation

Java 8:

- [guava](#)
- [guava-testlib](#)

Android / Java 7:

- [guava](#)
- [guava-testlib](#)

## Using Guava in your project

	Guava	Guava (Android)	Guava (GWT)
<b>Maven Identifier</b>	<a href="#">com.google.guava:guava:23.0</a>	<a href="#">com.google.guava:guava:23.0-android</a>	<a href="#">com.google.guava:guava-gwt:23.0</a>
<b>Jar</b>	<a href="#">guava-23.0.jar</a>	<a href="#">guava-23.0-android.jar</a>	<a href="#">guava-gwt-23.0.jar</a>
<b>Javadoc</b>	<a href="#">guava-23.0-javadoc.jar</a>	<a href="#">guava-23.0-android-javadoc.jar</a>	<a href="#">guava-gwt-23.0-javadoc.jar</a>
<b>Sources</b>	<a href="#">guava-23.0-sources.jar</a>	<a href="#">guava-23.0-android-sources.jar</a>	<a href="#">guava-gwt-23.0-sources.jar</a>

See [\[\[UseGuavaInYourBuild\]\]](#) for help integrating Guava into your build environment.

## API Changes

- Java 8: [Full JDiff Report](#) of changes since release 22.0.
- Android: [Full JDiff Report](#) of changes since release 22.0.

## Significant API additions and changes

### common.collect

- `ContiguousSet` : new convenience methods for closed or closed-open sets of `Integer` s or `Long` s
- `Set<Set<E>>` `Sets.combinations(Set<E> set, int size)` : returns all subsets of the given set that have the given size

### common.graph

- New types: `SuccessorsFunction` / `PredecessorsFunction`
  - These interfaces are each supertypes of `Graph` / `ValueGraph` / `Network` . They have a few purposes:
    - scaffolding for migrating the capabilities of `TreeTraverser` into `common.graph`

- facilitating users using their own graph data structures (when they don't need the full `common.graph` API)
- New methods on `Network` for the case when there is known to be at most one edge connecting two nodes
  - `java.util.Optional<E> edgeConnecting(N nodeU, N nodeV)` : Java 8 version only
  - `E edgeConnectingOrNull(N nodeU, N nodeV)` : all versions
- New method for `Graph` / `ValueGraph` / `Network`
  - `boolean hasEdgeConnecting(N nodeU, N nodeV)` : this is the preferred mechanism for determining whether there is an edge that connects two nodes
- Changes to methods for accessing edge values in `ValueGraph`
  - old method: `V edgeValue(N nodeU, N nodeV)` , throws `IllegalArgumentException` if no such edge exists
  - new method (Java 8 only): `java.util.Optional<V> edgeValue(N nodeU, N nodeV)` , returns `Optional.empty()` if no such edge exists
  - (no change to `@Nullable V edgeValueOrDefault(N nodeU, N nodeV, @Nullable V defaultValue)` ) (Java 7 & 8)

## common.hash

- `BloomFilter`
  - is now thread-safe
  - added `Collector` s for creating a `BloomFilter` from the contents of a `Stream`
- `PrimitiveSink` , `Hasher` and `HashFunction` : methods added to all for putting/hashing `ByteBuffer` s

## common.util.concurrent

- New type: `FluentFuture`
  - A `ListenableFuture` that supports fluent chains of operations
- `AbstractFuture` has been retrofitted to extend the new `FluentFuture` type. If you subclass `AbstractFuture` to add fluent methods like `transform()` , this may break you. If so, please [report the problem](#).
- Futures : new methods `scheduleAsync(AsyncCallable, long, TimeUnit, ScheduledExecutorService)` and `submitAsync(AsyncCallable, Executor)`