After auto-configuration has created the object of Resilience4JCircuitBreakerFactory, the ioc container takes that factory object and passes it as an input to the Customizer<Resilience4JCircuitBreakerFactory> and invokes customize method asking us to add our own configurations into it.

```
@Component
class Resilience4JCircuitBreakerFactoryCustomizer implements
Customizer<Resilience4JCircuitBreakerFactory> {
  void customize(Resilience4JCircuitBreakerFactory factory) {

    factory.configure(new Consumer<Resilience4JConfigBuilder>() {
      void accept(Resilience4JConfigBuilder builder) {

      }
    }, “distributornetworkconfig”);

    factory.configure(new Consumer<Resilience4JConfigBuilder>() {
      void accept(Resilience4JConfigBuilder builder) {

      }
    }, “inventorymgmtconfig”);

  }
}
```

Inside the Customizer class, in general we need to populate the Configuration objects into the CircuitBreakerFactory by binding an id. But if we create Configuration objects and populate into the CircuitBreakerFactory, it is going to waste the JVM memory because Factory will not use those configurations until we call

```
CircuitBreaker cb = factory.create(“id”);
```

So dont pre-instantiate the Configuration objects and populate them into CircuitBreakerFactory, rather the logic for creating the ConfigBuilder with configuration write it in an Consumer class and pass that object as an input to the factory asking to use for creating the CircuitBreaker object of that id.

```
class DistributorNetworkConfigConsumer implements
Consumer<Resilience4JConfigBuilder> {
  void accept(Resilience4JConfigBuilder builder) {
    builder.failureThresholdCount(30)
    .waitDurationInOpenState(Duration.ofSeconds(2));
  }
}

void customize(… factory) {
  factory.configure(new
      DistributorNetworkConfigConsumer(),
      “distributorNetworkConfig”);
}


factory.create(“distributorNetworkConfig”);


class Resilience4JCircuitBreakerFactory {
  Map<String, Consumer<Resilience4JConfigBuilder> map;

  Resilience4JCircuitBreaker create(String id) {
    Consumer<Resilience4JConfigBuilder> consumer = map.get(id);

    Resilience4JConfigBuilder builder = new
Resilience4JConfigBuilder();
    consumer.accept(builder);
    Resilience4JConfiguration config = builder.build();

    Resilience4JCircuitBreaker cb = new
Resilience4JCircuitBreaker(config);
    return cb;
  }
}
```