Rate limiter strategy **CHEAT SHEET**

Basics

This proactive resilience strategy **allows you to control the quantity of the inbound and/or the outbound invocations**.

You can configure the behaviour of the strategy via the subclass of **RateLimiter** or via **RateLimiterStrategyOptions /ConcurrencyLimiterOptions** object.

If the predefined threshold is exceeded, then it will throw an **RateLimiterRejectedException**.

Use the rate limiter to limit the incoming requests during a given period (also called a *window*).
Use the concurrency limiter to limit the outgoing concurrent request count against a resource.

This strategy is a thin wrapper over the **System.Threading.RateLimiting** APIs. Check out this article to further details.

Add the Polly.RateLimiting package to your project to be able to use this strategy.

Specify concurrency limit with bounded queue

```
new ResiliencePipelineBuilder()
  .AddConcurrencyLimiter(new ConcurrencyLimiterOptions()
  {
    PermitLimit = 10,
    QueueLimit = 20
  })
```

Specify a concurrency limiter + notification

```
new ResiliencePipelineBuilder()
  .AddRateLimiter(new RateLimiterStrategyOptions()
  {
    DefaultRateLimiterOptions = new ConcurrencyLimiterOptions
    {
      PermitLimit = 10,
      QueueLimit = 20
    },
    OnRejected = async args => await NotifyAsync(args.Context)
  })
```

Specify a sliding window rate limiter

```
new ResiliencePipelineBuilder()
  .AddRateLimiter(new SlidingWindowRateLimiter(
    new SlidingWindowRateLimiterOptions
    {
      PermitLimit = 100,
      Window = TimeSpan.FromMinutes(1)
    }))
```

Specify a fixed window rate limiter

```
new ResiliencePipelineBuilder()
  .AddRateLimiter(new FixedWindowRateLimiter(
    new FixedWindowRateLimiterOptions
    {
      PermitLimit = 100,
      QueueLimit = 500,
      QueueProcessingOrder = QueueProcessingOrder.OldestFirst,
      Window = TimeSpan.FromMinutes(1)
    })
```