# Request Units in DocumentDB

This article provides an overview of request units in [Microsoft Azure DocumentDB](https://azure.microsoft.com/services/documentdb/).

After reading this article, you’ll be able to answer the following questions:

* What are request units and request charges?
* How do I specify request unit capacity for a collection?
* How do I estimate my application’s request unit needs?
* What happens if I exceed request unit capacity for a collection?

## Request units and request charges

DocumentDB delivers fast, predictable performance by *reserving* resources to satisfy your application’s throughput needs. Because application load and access patterns change over time, DocumentDB allows you to easily increase or decrease the amount of reserved throughput available to your application.

With DocumentDB, reserved throughput is specified in terms of request units processing per second. You can think of request units as throughput currency, whereby you *reserve* an amount of guaranteed request units available to your application on per second basis. Each operation in DocumentDB - writing a document, performing a query, updating a document - consumes CPU, memory, and IOPS. That is, each operation incurs a *request charge*, which is expressed in *request units*. Understanding the factors which impact request unit charges, along with your application’s throughput requirements, enables you to run your application as cost effectively as possible.

## Specifying request unit capacity

When creating a DocumentDB collection, you specify the number of request units per second (RUs) you want reserved for the collection. Once the collection is created, the full allocation of RUs specified is reserved for the collection’s use. Each collection is guaranteed to have dedicated and isolated throughput characteristics.

It is important to note that DocumentDB operates on a reservation model; that is, you are billed for the amount of throughput *reserved* for the collection, regardless of how much of that throughput is actively *used*. Keep in mind, however, that as your application’s load, data, and usage patterns change you can easily scale up and down the amount of reserved RUs through DocumentDB SDKs or using the [Azure Portal](https://portal.azure.com). For more information on to scale throughput up and down, see [DocumentDB performance levels](/documentation/articles/documentdb-performance-levels).

## Request unit considerations

When estimating the number of request units to reserve for your DocumentDB collection, it is important to take the following variables into consideration:

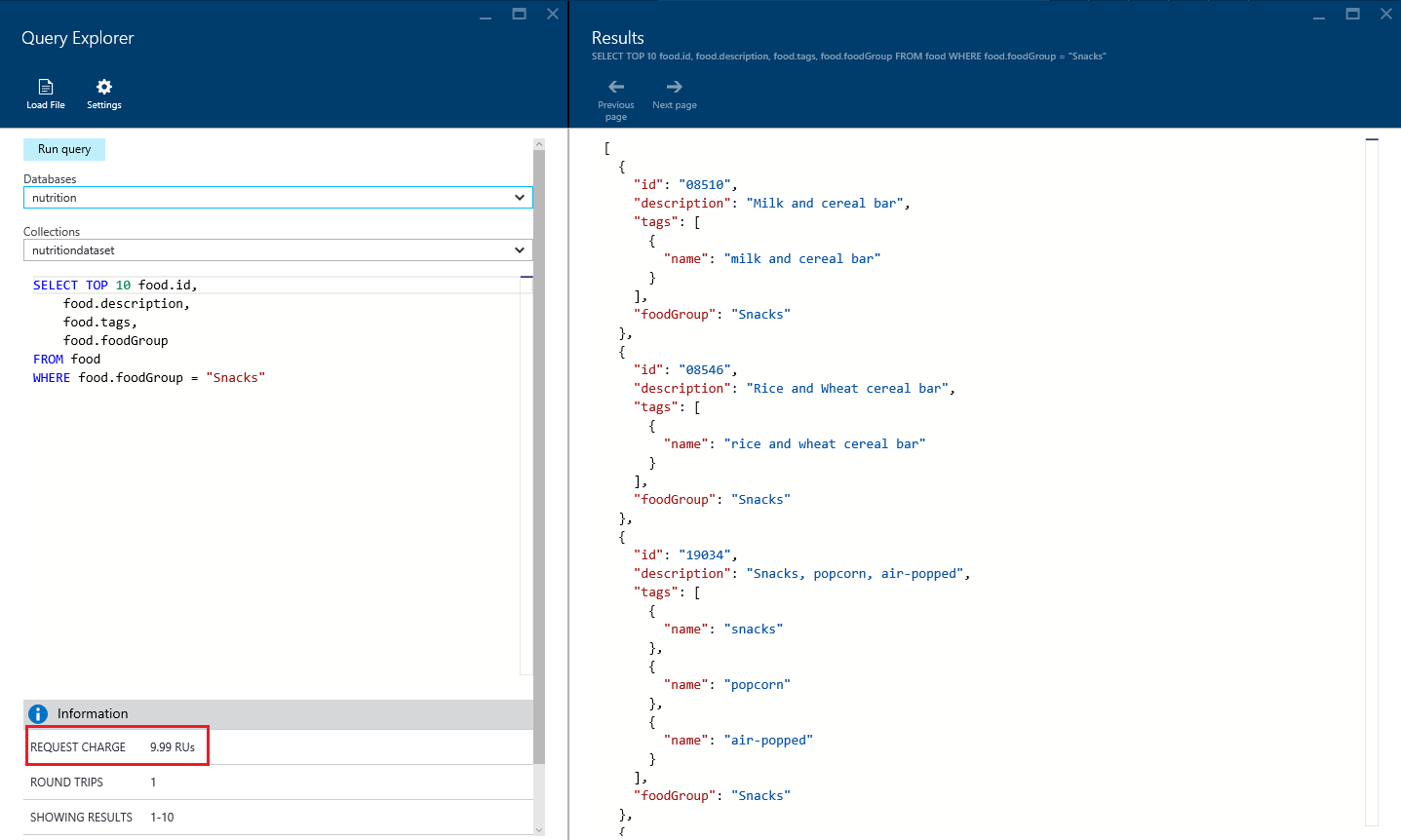
* **Document size**. As document sizes increase the units consumed to read or write the data will also increase.
* **Document property count**. Assuming default indexing of all properties, the units consumed to write a document will increase as the property count increases.
* **Data consistency**. When using data consistency levels of Strong or Bounded Staleness, additional units will be consumed to read documents.
* **Indexed properties**. An index policy on each collection determines which properties are indexed by default. You can reduce your request unit consumption by limiting the number of indexed properties or by enabling lazy indexing.
* **Document indexing**. By default each document is automatically indexed, you will consume fewer request units if you choose not to index some of your documents.
* **Query patterns**. The complexity of a query impacts how many Request Units are consumed for an operation. The number of predicates, nature of the predicates, projections, number of UDFs, and the size of the source data set all influence the cost of query operations.
* **Script usage**. As with queries, stored procedures and triggers consume request units based on the complexity of the operations being performed. As you develop your application, inspect the request charge header to better understand how each operation is consuming request unit capacity.

## Estimating throughput needs

A request unit is a normalized measure of request processing cost. A single request unit represents the processing capacity required to read (via self link or id) a single 1KB JSON document consisting of 10 unique property values (excluding system properties). A request to create (insert), replace or delete the same document will consume more processing from the service and thereby more request units.

[AZURE.NOTE] The baseline of 1 request unit for a 1KB document corresponds to a simple GET by self link or id of the document.

Every response from the DocumentDB service includes a custom header (x-ms-request-charge) that contains the request units consumed for the request. This header is also accessible through the DocumentDB SDKs. In the .NET SDK, RequestCharge is a property of the ResourceResponse object. For queries, the DocumentDB Query Explorer in the Azure portal provides request charge information for executed queries.



Examining RU charges in the Query Explorer

With this in mind, one method for estimating the amount of reserved throughput required by your application is to record the request unit charge associated with running typical operations against a representative document used by your application and then estimating the number of operations you anticipate performing each second. Be sure to measure and include typical queries and DocumentDB script usage as well.

[AZURE.NOTE]If you have document types which will differ dramatically in terms of size and the number of indexed properties, then record the applicable operation request unit charge associated with each *type* of typical document.

For example:

1. Record the request unit charge of creating (inserting) a typical document.
2. Record the request unit charge of reading a typical document.
3. Record the request unit charge of updating a typical document.
4. Record the request unit charge of typical, common document queries.
5. Record the request unit charge of any custom scripts (stored procedures, triggers, user-defined functions) leveraged by the application
6. Calculate the required request units given the estimated number of operations you anticipate to run each second.

## A request unit estimation example

Consider the following ~1KB document:

{  
 "id": "08259",  
"description": "Cereals ready-to-eat, KELLOGG, KELLOGG'S CRISPIX",  
"tags": [  
 {  
 "name": "cereals ready-to-eat"  
 },  
 {  
 "name": "kellogg"  
 },  
 {  
 "name": "kellogg's crispix"  
 }  
],  
"version": 1,  
"commonName": "Includes USDA Commodity B855",  
"manufacturerName": "Kellogg, Co.",  
"isFromSurvey": false,  
"foodGroup": "Breakfast Cereals",  
"nutrients": [  
 {  
 "id": "262",  
 "description": "Caffeine",  
 "nutritionValue": 0,  
 "units": "mg"  
 },  
 {  
 "id": "307",  
 "description": "Sodium, Na",  
 "nutritionValue": 611,  
 "units": "mg"  
 },  
 {  
 "id": "309",  
 "description": "Zinc, Zn",  
 "nutritionValue": 5.2,  
 "units": "mg"  
 }  
],  
"servings": [  
 {  
 "amount": 1,  
 "description": "cup (1 NLEA serving)",  
 "weightInGrams": 29  
 }  
]  
}

[AZURE.NOTE]Documents are minified in DocumentDB, so the system calculated size of the document above is slightly less than 1KB.

The following table shows approximate request unit charges for typical operations on this document (the approximate request unit charge assumes that the account consistency level is set to “Session” and that all documents are automatically indexed):

|  |  |
| --- | --- |
| Operation | Request Unit Charge |
| Create document | ~15 RU |
| Read document | ~1 RU |
| Query document by id | ~2.5 RU |

Additionally, this table shows approximate request unit charges for typical queries used in the application:

|  |  |  |
| --- | --- | --- |
| Query | Request Unit Charge | # of Returned Documents |
| Select food by id | ~2.5 RU | 1 |
| Select foods by manufacturer | ~7 RU | 7 |
| Select by food group and order by weight | ~70 RU | 100 |
| Select top 10 foods in a food group | ~10 RU | 10 |

[AZURE.NOTE]RU charges vary based on the number of documents returned.

With this information, we can estimate the RU requirements for this application given the number of operations and queries we expect per second:

|  |  |  |
| --- | --- | --- |
| Operation/Query | Estimated number per second | Required RUs |
| Create document | 10 | 150 |
| Read document | 100 | 100 |
| Select foods by manufacturer | 25 | 175 |
| Select by food group | 10 | 700 |
| Select top 10 | 15 | 150 Total |

In this case, we expect an average throughput requirement of 1,275 RU/s. Rounding up to the nearest 100, we would provision 1,300 RU/s for this application’s collection.

## Exceeding reserved throughput limits

Recall that request unit consumption is evaluated as a rate per second. For applications that exceed the provisioned request unit rate for a collection, requests to that collection will be throttled until the rate drops below the reserved level. When a throttle occurs, the server will preemptively end the request with RequestRateTooLarge (HTTP status code 429) and return the x-ms-retry-after-ms header indicating the amount of time, in milliseconds, that the user must wait before reattempting the request.

HTTP Status 429  
Status Line: RequestRateTooLarge  
x-ms-retry-after-ms :100

If you are using the .NET Client SDK and LINQ queries, then most of the time you never have to deal with this exception, as the current version of the .NET Client SDK implicitly catches this response, respects the server-specified retry-after header, and retries the request. Unless your account is being accessed concurrently by multiple clients, the next retry will succeed.

If you have more than one client cumulatively operating above the request rate, the default retry behavior may not suffice, and the client will throw a DocumentClientException with status code 429 to the application. In cases such as this, you may consider handling retry behavior and logic in your application’s error handling routines or increasing the reserved throughput for the collection.

## Next steps

To learn more about reserved throughput with Azure DocumentDB, explore these resources:

* [DocumentDB pricing](https://azure.microsoft.com/pricing/details/documentdb/)
* [Managing DocumentDB capacity](/documentation/articles/documentdb-manage)
* [Modeling data in DocumentDB](/documentation/articles/documentdb-modeling-data)
* [DocumentDB performance levels](/documentation/articles/documentdb-partition-data)

To learn more about DocumentDB, see the Azure DocumentDB [documentation](https://azure.microsoft.com/documentation/services/documentdb/).