Azure - Queues

In the common language used by developers, a queue is a data structure used to store data which follows First in-First out rule. A data item can be inserted from back of the queue while it is retrieved from front. Azure queues are a very similar concept that is used to store the messages in a queue. A sender sends the message and a client receives and processes them. A message has few attributes attached to it, for example expiry time.

A client usually processes and deletes the message. Windows Azure service lets the message to be stored for 7 days and later it gets deleted automatically, if it is not deleted by the client. There can be one sender and one client or one sender and many clients or many sender and many clients.

There are two services offered by Windows Azure for message queues. This chapter covers Windows Azure queue. The other service is called ‘Service Bus queue’.

Decoupling the components is one of the advantages of message queue services. It runs in an asynchronous environment where messages can be sent among the different components of an application. Thus, it provides an efficient solution for managing workflows and tasks. For example, a message to complete a task is sent from the frontend of the application and is received by a backend worker, who then completes the task and deletes the message.

## Considerations

The messages in the storage queue are not replicated anywhere, that means there is only one copy of your message. The maximum number of messages that can be processed are 20,000. The maximum size of a message can be 40 kb.

## Managing Queues using PowerShell

### Create a Queue

**Step 1** − Right-click on Windows PowerShell in the taskbar. Choose ‘Run ISE as administrator’.

**Step 2** − Run the following command to access your account. Please replace the highlighted part for your account.

$context = New-AzureStorageContext -StorageAccountName tutorialspoint StorageAccountKey

iUZNeeJD+ChFHt9XHL6D5rkKFWjzyW4FhV0iLyvweDi+Xtzfy76juPzJ+mWtDmbqCWjsu/nr+1pqBJj rdOO2+A==

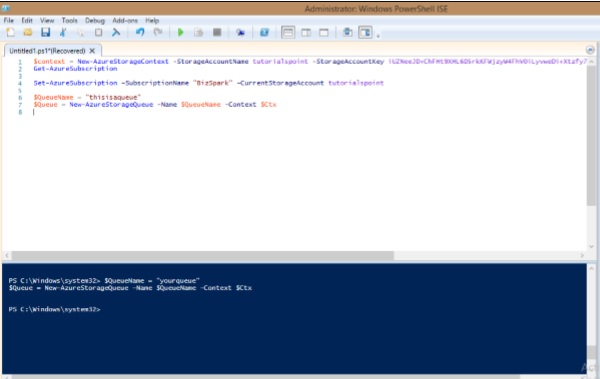
**Step 3** − Specify the storage account in which you want to create a queue.

Set-AzureSubscription –SubscriptionName "BizSpark" -CurrentStorageAccount tutorialspoint

**Step 4** − Create a Queue.

$QueueName = "thisisaqueue"

$Queue = New-AzureStorageQueue –Name $QueueName -Context $Ctx



### Retrieve a Queue

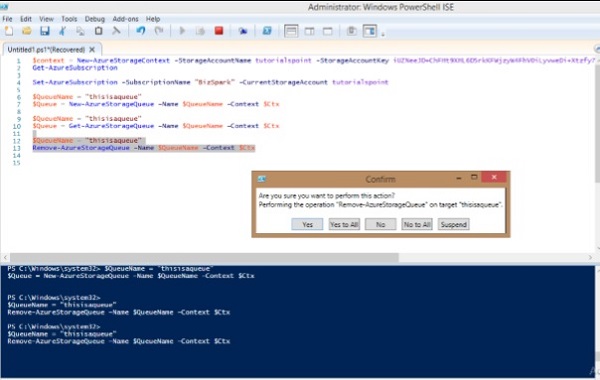
$QueueName = "thisisaqueue"

$Queue = Get-AzureStorageQueue –Name $QueueName –Context $Ctx

### Delete a Queue

$QueueName = "thisisaqueue"

Remove-AzureStorageQueue –Name $QueueName –Context $Ctx



### Insert a Message into a Queue

**Step 1** − Login to your account.

$context = New-AzureStorageContext -StorageAccountName tutorialspoint StorageAccountKey

iUZNeeJD+ChFHt9XHL6D5rkKFWjzyW4FhV0iLyvweDi+Xtzfy76juPzJ+mWtDmbqCWjsu/nr+1pqBJj rdOO2+A==

**Step 2** − Specify the storage account you want to use.

Set-AzureSubscription –SubscriptionName "BizSpark" -CurrentStorageAccount tutorialspoint

**Step 3** − Retrieve the queue and then insert the message.

$QueueName = "myqueue"

$Queue = Get-AzureStorageQueue -Name $QueueName -Context $ctx

if ($Queue -ne $null) {

$QueueMessage = New-Object -TypeName Microsoft.WindowsAzure.Storage.Queue.CloudQueueMessage

-ArgumentList "my message is this"

$Queue.CloudQueue.AddMessage($QueueMessage)

}

The ‘if’ condition in the script above checks if the queue specified exists or not.

### Dequeue Next Message from Queue

**Step 1** − First connect to your account and specify the storage account, by running the commands as shown in the above steps.

**Step 2** − Retrieve the queue.

$QueueName = "myqueue"

$Queue = Get-AzureStorageQueue -Name $QueueName -Context $ctx

$InvisibleTimeout = [System.TimeSpan]::FromSeconds(10)

**Step 3** − Dequeue the next message.

$QueueMessage = $Queue.CloudQueue.GetMessage($InvisibleTimeout)

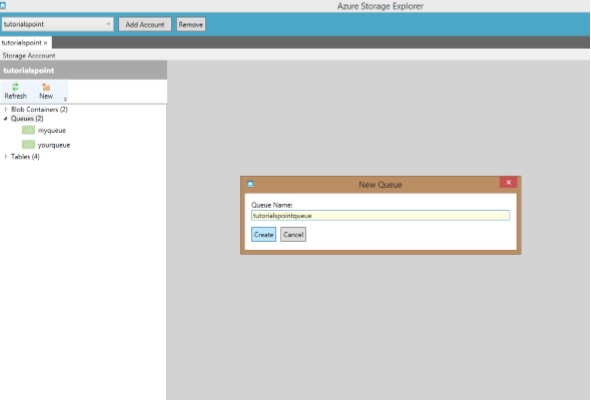
**Step 4** − Delete the dequeued message.

$Queue.CloudQueue.DeleteMessage($QueueMessage)

## Managing Queues using Azure Storage Explorer

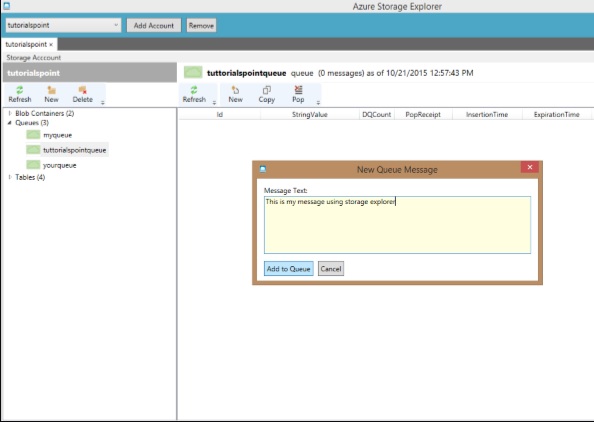
**Step 1** − Select the storage account from the dropdown at the top right. Accounts will be displayed if you have added them during your previous use. If not, you can add account and it will ask for your credentials. After signing in, you will be logged into your account in Azure Storage Explorer.

**Step 2** − You can add a new queue by selecting ‘Queues’ from the left panel and clicking ‘New’ as shown in the following image.



**Step 3** − Enter the name of Queue and it is created in your storage account.

**Step 4** − Add and delete the messages by selecting the queue in the left panel.



# Azure - Tables

Storing a table does not mean relational database here. Azure Storage can store just a table without any foreign keys or any other kind of relation. These tables are highly scalable and ideal for handling large amount of data. Tables can be stored and queried for large amount of data. The relational database can be stored using SQL Data Services, which is a separate service.

The three main parts of service are −

* Tables
* Entities
* Properties

For example, if ‘Book’ is an entity, its properties will be Id, Title, Publisher, Author etc. Table will be created for a collection of entities. There can be 252 custom properties and 3 system properties. An entity will always have system properties which are PartitionKey, RowKey and Timestamp. Timestamp is system generated but you will have to specify the PartitionKey and RowKey while inserting data into the table. The example below will make it clearer. Table name and Property name is case sensitive which should always be considered while creating a table.

## How to Manage Tables Using PowerShell

**Step 1** − Download and install Windows PowerShell as discussed previously in the tutorial.

**Step 2** − Right-click on ‘Windows PowerShell’, choose ‘Pin to Taskbar’ to pin it on the taskbar of your computer.

**Step 3** − Choose ‘Run ISE as Administrator’.

### Creating a Table

**Step 1** − Copy the following commands and paste into the screen. Replace the highlighted text with your account.

**Step 2** − Login into your account.

$StorageAccountName = "mystorageaccount"

$StorageAccountKey = "mystoragekey"

$Ctx = New-AzureStorageContext $StorageAccountName - StorageAccountKey

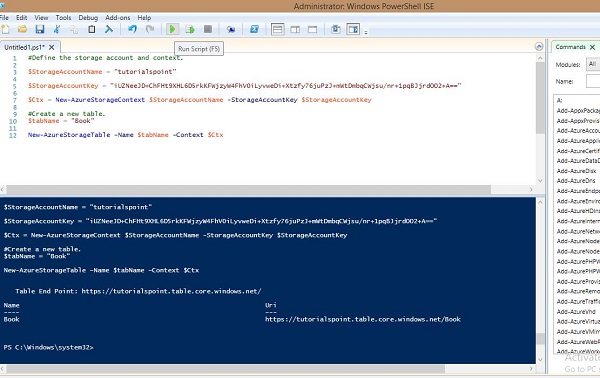
$StorageAccountKey

**Step 3** − Create a new table.

$tabName = "Mytablename"

New-AzureStorageTable –Name $tabName –Context $Ctx

The following image shows a table being created by the name of ‘book’.



You can see that it has given the following end point as a result.

https://tutorialspoint.table.core.windows.net/Book

Similarly, you can retrieve, delete and insert data into the table using preset commands in PowerShell.

### Retrieve Table

$tabName = "Book"

Get-AzureStorageTable –Name $tabName –Context $Ctx

### Delete Table

$tabName = "Book"

Remove-AzureStorageTable –Name $tabName –Context $Ctx

### Insert rows into Table

function Add-Entity() {

[CmdletBinding()]

param(

$table,

[String]$partitionKey,

[String]$rowKey,

[String]$title,

[Int]$id,

[String]$publisher,

[String]$author

)

$entity = New-Object -TypeName Microsoft.WindowsAzure.Storage.Table.DynamicTableEntity

-ArgumentList $partitionKey, $rowKey

$entity.Properties.Add("Title", $title)

$entity.Properties.Add("ID", $id)

$entity.Properties.Add("Publisher", $publisher)

$entity.Properties.Add("Author", $author)

$result = $table.CloudTable.Execute(

[Microsoft.WindowsAzure.Storage.Table.TableOperation]

::Insert($entity))

}

$StorageAccountName = "tutorialspoint"

$StorageAccountKey = Get-AzureStorageKey -StorageAccountName $StorageAccountName

$Ctx = New-AzureStorageContext $StorageAccountName - StorageAccountKey

$StorageAccountKey.Primary

$TableName = "Book"

$table = Get-AzureStorageTable –Name $TableName -Context $Ctx -ErrorAction Ignore

#Add multiple entities to a table.

Add-Entity -Table $table -PartitionKey Partition1 -RowKey Row1 -Title .Net -Id 1

-Publisher abc -Author abc

Add-Entity -Table $table -PartitionKey Partition2 -RowKey Row2 -Title JAVA -Id 2

-Publisher abc -Author abc

Add-Entity -Table $table -PartitionKey Partition3 -RowKey Row3 -Title PHP -Id 3

-Publisher xyz -Author xyz

Add-Entity -Table $table -PartitionKey Partition4 -RowKey Row4 -Title SQL -Id 4

-Publisher xyz -Author xyz

### Retrieve Table Data

$StorageAccountName = "tutorialspoint"

$StorageAccountKey = Get-AzureStorageKey - StorageAccountName $StorageAccountName

$Ctx = New-AzureStorageContext – StorageAccountName $StorageAccountName -

StorageAccountKey $StorageAccountKey.Primary;

$TableName = "Book"

#Get a reference to a table.

$table = Get-AzureStorageTable –Name $TableName -Context $Ctx

#Create a table query.

$query = New-Object Microsoft.WindowsAzure.Storage.Table.TableQuery

#Define columns to select.

$list = New-Object System.Collections.Generic.List[string]

$list.Add("RowKey")

$list.Add("ID")

$list.Add("Title")

$list.Add("Publisher")

$list.Add("Author")

#Set query details.

$query.FilterString = "ID gt 0"

$query.SelectColumns = $list

$query.TakeCount = 20

#Execute the query.

$entities = $table.CloudTable.ExecuteQuery($query)

#Display entity properties with the table format.

$entities | Format-Table PartitionKey, RowKey, @{ Label = "Title";

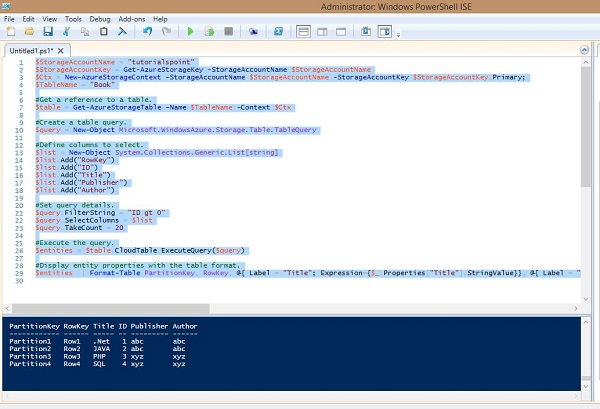
Expression={$\_.Properties["Title"].StringValue}}, @{ Label = "ID";

Expression={$\_.Properties[“ID”].Int32Value}}, @{ Label = "Publisher";

Expression={$\_.Properties[“Publisher”].StringValue}}, @{ Label = "Author";

Expression={$\_.Properties[“Author”].StringValue}} -AutoSize

The output will be as shown in the following image.



### Delete Rows from Table

$StorageAccountName = "tutorialspoint"

$StorageAccountKey = Get-AzureStorageKey - StorageAccountName $StorageAccountName

$Ctx = New-AzureStorageContext – StorageAccountName $StorageAccountName -

StorageAccountKey $StorageAccountKey.Primary

#Retrieve the table.

$TableName = "Book"

$table = Get-AzureStorageTable -Name $TableName -Context $Ctx -ErrorAction

Ignore

#If the table exists, start deleting its entities.

if ($table -ne $null) {

#Together the PartitionKey and RowKey uniquely identify every

#entity within a table.

$tableResult = $table.CloudTable.Execute(

[Microsoft.WindowsAzure.Storage.Table.TableOperation]

::Retrieve(“Partition1”, "Row1"))

$entity = $tableResult.Result;

if ($entity -ne $null) {

$table.CloudTable.Execute(

[Microsoft.WindowsAzure.Storage.Table.TableOperation]

::Delete($entity))

}

}

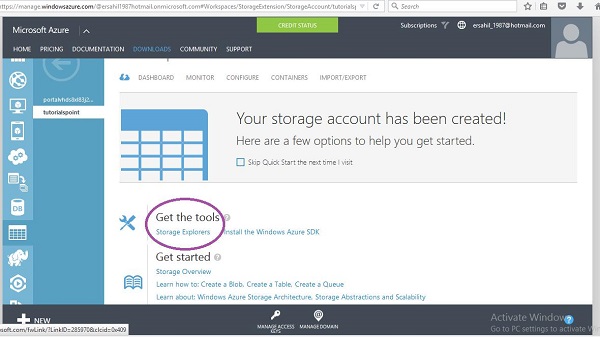
The above script will delete the first row from the table, as you can see that we have specified Partition1 and Row1 in the script. After you are done with deleting the row, you can check the result by running the script for retrieving rows. There you will see that the first row is deleted.

While running these commands please ensure that you have replaced the accountname with your account name, accountkey with your account key.

## How to Manage Table using Azure Storage Explorer

**Step 1** − Login in to your Azure account and go to your storage account.

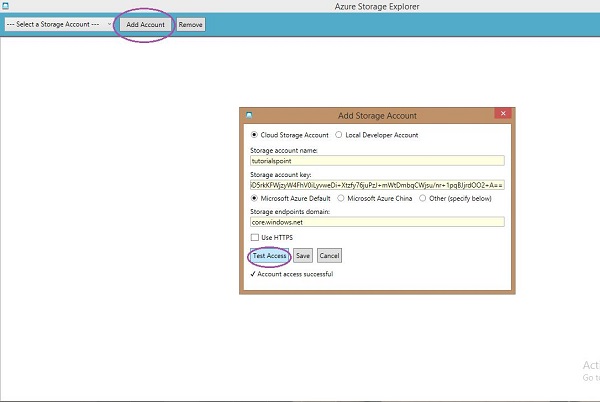
**Step 2** − Click on the link ‘Storage explorer’ as shown in purple circle in the following image.



**Step 3** − Choose ‘Azure Storage Explorer for Windows’ from the list. It is a free tool that you can download and install on your computer.

**Step 4** − Run this program on your computer and click ‘Add Account’ button at the top.

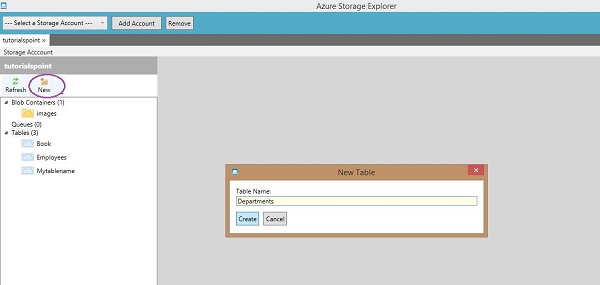
**Step 5** − Enter ‘Storage Account Name’ and ‘Storage account Key’ and click ‘Test Access. The buttons are encircled in following image.



**Step 6** − If you already have any tables in storage you will see in the left panel under ‘Tables’. You can see the rows by clicking on them.

### Create a Table

**Step 1** − Click on ‘New’ and enter the table name as shown in the following image.

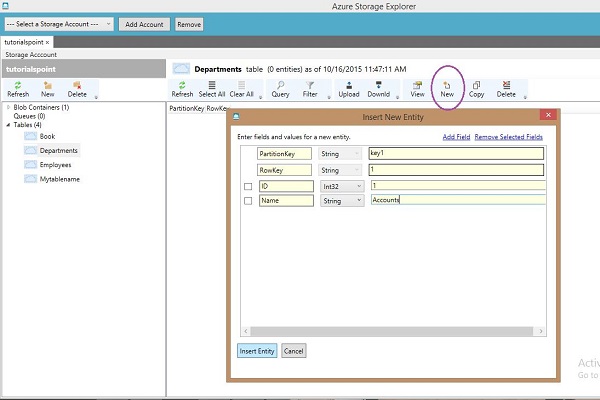


### Insert Row into Table

**Step 1** − Click on ‘New’.

**Step 2** − Enter Field Name.

**Step 3** − Select data type from dropdown and enter field value.



**Step 4** − To see the rows created click on the table name in the left panel.

Azure Storage Explorer is very basic and easy interface to manage tables. You can easily create, delete, upload, and download tables using this interface. This makes the tasks very easy for developers as compared to writing lengthy scripts in Windows PowerShell.