1. Install erlang

<http://www.erlang.org/downloads>

1. Install Rabbit

<https://www.rabbitmq.com/download.html>

1. En .NET el cliente

PM> Install-Package RabbitMQ.Client –Version 4.1.1

**"Hello World"**

Two programs in C#;

* A producer that sends a single message,
* A consumer that receives messages and prints them out.

In the diagram below, "P" is our producer and "C" is our consumer. The box in the middle is a queue - a message buffer that RabbitMQ keeps on behalf of the consumer.

(P) -> [|||] -> (C)

#### The .NET client library

RabbitMQ speaks multiple protocols. This tutorial uses AMQP 0-9-1, which is an open, general-purpose protocol for messaging. There are a number of clients for RabbitMQ in [many different languages](http://rabbitmq.com/devtools.html). We'll use the .NET client provided by RabbitMQ.

Download the [client library package](http://www.rabbitmq.com/dotnet.html), and check its signature as described. Extract it and copy "RabbitMQ.Client.dll" to your working folder.

You also need to ensure your system can find the C# compiler csc.exe, you may need to add ;C:\WINDOWS\Microsoft.NET\Framework\v3.5 (change .NET version to fit your installation) to your Path.

**Sending**

We'll call our message sender Send and our message receiver Receive.cs. The sender will connect to RabbitMQ, send a single message, then exit.

The connection abstracts the socket connection, and takes care of protocol version negotiation and authentication and so on for us. Here we connect to a broker on the local machine - hence the localhost.

Next we create a channel, which is where most of the API for getting things done resides.

To send, we must declare a queue for us to send to; then we can publish a message to the queue: **Send.cs**

using System;

using RabbitMQ.Client;

using System.Text;

class Send

{

public static void Main()

{

**// Connection to server**

var factory = new ConnectionFactory() { HostName = "localhost" };

using(var connection = factory.CreateConnection())

using(var channel = connection.CreateModel())

{

channel.QueueDeclare(queue: "hello",

durable: false,

exclusive: false,

autoDelete: false,

arguments: null);

**// MESSAGE ONE**

string message = "Hello World!";

var body = Encoding.UTF8.GetBytes(message);

channel.BasicPublish(exchange: "",

routingKey: "hello",

basicProperties: null,

body: body);

**// MESSAGE TWO**

message = "Hello Again";

body = Encoding.UTF8.GetBytes(message);

channel.BasicPublish(exchange: "",

routingKey: "hello",

basicProperties: null,

body: body);

//Console.WriteLine(" [x] Sent {0}", message);

}

//Console.WriteLine(" Press [enter] to exit.");

//Console.ReadLine();

}

}

Declaring a queue is idempotent - it will only be created if it doesn't exist already. The message content is a byte array, so you can encode whatever you like there.

When the code above finishes running, the channel and the connection will be disposed.

#### Sending doesn't work!

1. Maybe the broker was started without enough free disk space (by default it needs at least 50 MB free). Check the broker logfile to confirm and reduce the limit if necessary. The [configuration file documentation](http://www.rabbitmq.com/configure.html#config-items) will show you how to set disk\_free\_limit.
2. Not connecting to localhost

**Receiving**

Our receiver is pushed messages from RabbitMQ, so unlike the sender which publishes a single message, we'll keep it running to listen for messages and print them out.

The code (in [**Receive.cs**](https://github.com/rabbitmq/rabbitmq-tutorials/blob/master/dotnet/Receive/Receive.cs)) has almost the same using statements as Send:

Setting up is the same as the sender; we open a connection and a channel, and declare the queue from which we're going to consume. Note this matches up with the queue that send publishes to.

We might start the receiver before the sender, we want to make sure the queue exists before we try to consume messages from it.

We're about to tell the server to deliver us the messages from the queue. Since it will push us messages asynchronously, we provide a callback. That is what

**EventingBasicConsumer.Received** event handler does.

using RabbitMQ.Client;

using RabbitMQ.Client.Events;

using System;

using System.Text;

class Receive

{

public static void Main()

{

**// Connection**

var factory = new ConnectionFactory() { HostName = "localhost" };

using(var connection = factory.CreateConnection())

using(var channel = connection.CreateModel())

{

channel.QueueDeclare(queue: "hello",

durable: false,

exclusive: false,

autoDelete: false,

arguments: null);

var consumer = new EventingBasicConsumer(channel);

consumer.Received += (model, ea) =>

{

var body = ea.Body;

var message = Encoding.UTF8.GetString(body);

// Console.WriteLine(" [x] Received {0}", message);

MessageBox.Show(message);

};

channel.BasicConsume(queue: "hello",

noAck: true,

consumer: consumer);

//Console.WriteLine(" Press [enter] to exit.");

//Console.ReadLine();

}

}

}

You can compile both of these by referencing the RabbitMQ .NET client assembly. We're using the command line (cmd.exe and csc) to compile and run the code. Alternatively you could use Visual Studio.

$ csc /r:"RabbitMQ.Client.dll" Send.cs

$ csc /r:"RabbitMQ.Client.dll" Receive.cs

Then run the executable

$ Send.exe

then, run the receiver:

$ Receive.exe

The receiver will print the message it gets from the sender via RabbitMQ. The receiver will keep running, waiting for messages (Use Ctrl-C to stop it), so try running the sender from another terminal.

If you want to check on the queue, try using rabbitmqctl list\_queues.

Hello World!

Time to move on to [part 2](https://www.rabbitmq.com/tutorials/tutorial-two-dotnet.html) and build a simple work queue.

**SERVER LISTENING**

using System;

using System.Windows.Forms;

using RabbitMQ.Client;

using RabbitMQ.Client.Events;

using System.Text;

using System.Threading;

namespace WindowsFormsApplication2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

var factory = new ConnectionFactory() { HostName = "localhost" };

using (var connection = factory.CreateConnection())

using (var channel = connection.CreateModel())

{

channel.QueueDeclare(queue: "hello",

durable: false,

exclusive: false,

autoDelete: false,

arguments: null);

while (true)

{

// RECEIVED

var consumer = new EventingBasicConsumer(channel);

consumer.Received += (model, ea) =>

{

var body = ea.Body;

var message = Encoding.UTF8.GetString(body);

MessageBox.Show(message);

};

channel.BasicConsume(queue: "hello",

noAck: true,

consumer: consumer);

}

Thread.Sleep(2000);

}

}

}

}

**CLICK BUTTON AND SEND MSG**

using System;

using System.Windows.Forms;

using RabbitMQ.Client;

namespace WindowsFormsApplication3

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

// Connection to server

var factory = new ConnectionFactory() { HostName = "localhost" };

using (var connection = factory.CreateConnection())

using (var channel = connection.CreateModel())

{

channel.QueueDeclare(queue: "hello",

durable: false,

exclusive: false,

autoDelete: false,

arguments: null);

// MESSAGE ONE

string message = textBox1.Text;

var body = System.Text.Encoding.UTF8.GetBytes(message);

channel.BasicPublish(exchange: "",

routingKey: "hello",

basicProperties: null,

body: body);

MessageBox.Show("message sent");

//Console.WriteLine(" [x] Sent {0}", message);

}

}

}

}