Azure服务总线API调用示例

编程调用Azure服务总线的相关服务，微软有提供基于.NET的dll（[Nuget](https://www.nuget.org/packages/WindowsAzure.ServiceBus/)），也有基于Java的Jar包（[Mvnrepository](https://mvnrepository.com/artifact/com.microsoft.azure/azure-servicebus)），另外也可以直接调用服务总线API。这片文章就提供Azure服务总线API调用示例，以供参考。主要包含以下内容：

1. 如何从策略生成共享访问令牌
2. 调用API发送消息到服务总线队列
3. 调用API从服务总线队列接收消息，包括ReceiveAndDelete模式和PeekLock模式。

## 如何从策略生成共享访问令牌

当调用服务总线API时，必须要提供共享访问令牌（SAS Token）作为Authorization头部，以作认证。那么如何生成这个SAS Token呢？

SAS Token的具体构成是这样的：SharedAccessSignature sr={资源地址}&sig={签名}&se={过期时间}&skn={策略名称}。

* 资源地址：要访问资源的URL地址，比如demo命名空间下的q1队列https://demo.servicebus.chinacloudapi.cn/q1
* 签名：对由资源地址，换行符和过期时间组成的字符串用策略秘钥通过SHA-256 哈希后生成的Base64String。
* 过期时间：自纪元算起，以秒为单位。
* 策略名称：相应策略的名称。

|  |
| --- |
| static string createToken(string resourceUri, string keyName, string key)  {  TimeSpan sinceEpoch = DateTime.UtcNow - new DateTime(1970, 1, 1);  var week = 60 \* 60 \* 24 \* 7;  var expiry = Convert.ToString((int)sinceEpoch.TotalSeconds + week);  string stringToSign = HttpUtility.UrlEncode(resourceUri) + "\n" + expiry;  HMACSHA256 hmac = new HMACSHA256(Encoding.UTF8.GetBytes(key));  var signature = Convert.ToBase64String(hmac.ComputeHash(Encoding.UTF8.GetBytes(stringToSign)));  var sasToken = String.Format(CultureInfo.InvariantCulture,  "SharedAccessSignature sr={0}&sig={1}&se={2}&skn={3}",  HttpUtility.UrlEncode(resourceUri),  HttpUtility.UrlEncode(signature),  expiry,  keyName);  return sasToken;  } |

## 调用API发送消息到服务总线队列

发送消息的API具体描述可参照官方文档：<https://docs.microsoft.com/en-us/rest/api/servicebus/send-message-to-queue>

最关键的就是将HTTP请求的Authorization头部的值设置为根据拥有发送权限策略生成的SAS Token。

|  |
| --- |
| static async Task SendMessageAsync(ServiceBusHttpMessage message, string token)  {  var address = $"https://{serviceBusNamespace}.servicebus.chinacloudapi.cn/{queueName}/messages";  HttpContent postContent = new ByteArrayContent(message.Body);  // Serialize BrokerProperties.  var brokerProps = JsonConvert.SerializeObject(message.SystemProperties,  Formatting.None,  new JsonSerializerSettings { NullValueHandling = NullValueHandling.Ignore,  DefaultValueHandling = DefaultValueHandling.Ignore });  postContent.Headers.Add("BrokerProperties", brokerProps);  // Add custom properties.  foreach (string key in message.CustomProperties)  {  postContent.Headers.Add(key, message.CustomProperties.GetValues(key));  }  var httpClient = new HttpClient();  httpClient.DefaultRequestHeaders.Add("Authorization", token);  httpClient.DefaultRequestHeaders.Add("ContentType", "application/atom+xml;type=entry;charset=utf-8");  // Send message.  HttpResponseMessage response = null;  try  {  response = await httpClient.PostAsync($"{address}?timeout=60", postContent);  response.EnsureSuccessStatusCode();  Console.WriteLine("SendMessage successfully!");  }  catch (HttpRequestException ex)  {  Console.WriteLine($"SendMessage failed: {ex.Message}");  }  response.Dispose();  } |

## 调用API从服务总线队列接收消息

ReceiveAndDelete模式API：<https://docs.microsoft.com/en-us/rest/api/servicebus/receive-and-delete-message-destructive-read>

PeekLock模式API：<https://docs.microsoft.com/en-us/rest/api/servicebus/peek-lock-message-non-destructive-read>

两者的区别在于，ReceiveAndDelete模式消息被接收后就自动从队列中删除了，而PeekLock模式只是把消息锁住，然后由客户端来删除。所以两种模式的API URI是一致的，只是HTTP方法不一样，前者是DELETE，后者是POST。

|  |
| --- |
| static async Task<ServiceBusHttpMessage> ReceiveMessageAsync(string token, bool deleteMessage = true)  {  var address = $"https://{serviceBusNamespace}.servicebus.chinacloudapi.cn/{queueName}/messages/head";  var httpClient = new HttpClient();  httpClient.DefaultRequestHeaders.Add("Authorization", token);  HttpResponseMessage response = null;  try  {  if (deleteMessage)  {  // receive and delete  response = await httpClient.DeleteAsync($"{address}?timeout=60");  }  else  {  // peek lock  response = await httpClient.PostAsync($"{address}?timeout=60", new ByteArrayContent(new Byte[0]));  }    response.EnsureSuccessStatusCode();  Console.WriteLine($"ReceiveMessage successfully!");  }  catch (HttpRequestException ex)  {  Console.WriteLine($"ReceiveMessage failed: {ex.Message}");  return null;  }  var message = await ResolveMessageFromResponse(response);  response.Dispose();  return message;  } |

完整代码：<https://github.com/allenhula/azure-china-get-started/tree/master/ServiceBus/CSharp/SBRestApiDemo>

## 更多资源：

从策略生成共享访问令牌（SAS Token）：<https://www.azure.cn/documentation/articles/service-bus-sas-overview/#bookmark-2>。

服务总线REST API：<https://docs.microsoft.com/en-us/rest/api/servicebus/service-bus-runtime-rest>