EventHubClient “C” Library

Revision 0.4

5/20/2015

# Revisions:

|  |  |  |
| --- | --- | --- |
| Revision | Updated By | Major updates |
| 0.1 | Tameraw | Initial draft |
| 0.2 | Tameraw | Updated descriptions & some APIs based on initial comments |
| 0.3 | Tameraw | More updates around the APIs. |
| 0.4 | Dcristo | Updated Async APIs |

# Overview

The EventHubClient “C” library offers developers a means of connecting to an already created EventHub and the ability to send data to that EventHub. The library offers the following features:

* The library communicates to an existing EventHub over AMQP protocol.
* The library uses Proton-C to establish the AMQP communication necessary
* The output is a static lib.
* The library offers the following APIs:
  + EventHubClientLib\_CreateFromConnectionString
  + EventHubClientLib\_Send
  + EventHubClientLib\_SendAsync
  + EventHubClientLib\_SendBatch
  + EventHubClientLib\_SendBatchAsync
  + EventHubClientLib\_Destroy

# Reference

[EventHubClient Class for .net](http://msdn.microsoft.com/en-us/library/microsoft.servicebus.messaging.eventhubclient.aspx)

# Types defined by the EventHubClient module:

typedef enum EVENTHUBCLIENT\_RESULT\_VALUES\_TAG

{

    EVENTHUBCLIENT\_OK,

    EVENTHUBCLIENT\_INVALID\_ARG,

    EVENTHUBCLIENT\_INVALID\_CONNECTION\_STRING,

    EVENTHUBCLIENT\_FAILED

} EVENTHUBCLIENT\_RESULT\_VALUES;

typedef void\* EVENTHUBCLIENT\_HANDLE;

typedef void (\*EventHubClient\_CallBack)(EVENTHUBCLIENT\_HANDLE clientHandle, void\* context);

# EventHubClient APIs

## EVENTHUBCLIENT\_HANDLE EventHubClient\_CreateFromConnectionString(const char\* connectionString, const char\* eventHubPath)

Initializes the EventHubClient internal structures.

### Arguments:

* *connectionString* – The service bus connection string to be used.
* *eventHubPath* – The path to the Event Hub to connect to.

The connectionString is expected to be in the following format:

Endpoint=sb://[**namespace**].servicebus.windows.net/;SharedAccessKeyName=[**key name**];SharedAccessKey=[**key value**]

The user can get to this connection string from the Azure Portal under Service Bus.

### Returns:

* None-NULL handle value that is used when invoking other functions EventHubClient.
* NULL on failure.

### Example:

#include "EventHubClient.h"

#include "EventData.h"

int main(int argc, char\*\* argv)

{

    const char\* connectionString = "Endpoint=sb://xxx.servicebus.windows.net/;SharedAccessKeyName=Device0SendAccessKey;SharedAccessKey=xyztdyfjhdhejwldif12bogus==";

    const char\* path = "tamer-eh-02";

    const char\* msgText = "Hello From EventHubClient";

    printf("Starting the EventHubClientSample...\n");

    EVENTHUBCLIENT\_HANDLE clientHandle = EventHubClient\_CreateFromConnectionString(connectionString, path);

    if (clientHandle == NULL)

    {

        /\* error \*/

    }

    else

    {

        /\* do some cool stuff here :) \*/

    }

    EventHubClient\_Destroy(clientHandle);

    return 0;

}

## void EventHubClient\_Destroy(EVENTHUBCLIENT\_HANDLE eventHubHandle)

Disposes of resources allocated by the EventHubClient module.

### Arguments:

* *clientHandle* – The handle created by a call to the EventHubClient\_CreateFromConnectionString.

### Returns:

* None

### Example:

#include "EventHubClient.h"

int main(int argc, char\*\* argv)

{

. . .

    EVENTHUBCLIENT\_HANDLE clientHandle = EventHubClient\_CreateFromConnectionString(connectionString, path);

. . .

    EventHubClient\_Destroy(clientHandle);

    return 0;

}

## EVENTHUBCLIENT\_RESULT EventHubClient\_Send(EVENTHUBCLIENT\_HANDLE clientHandle, EVENTDATA\_HANDLE eventDataHandle)

Synchronously sends one EventData to an existing EventHub.

### Arguments:

* *clientHandle* – Handle created by a call to the EventHubClient\_CreateFromConnectionString.
* *eventDataHandle –* Handle created by a call to the EventData\_Create which encapsulates the data to be sent to EventHub.

### Returns:

* EVENTHUBCLIENT\_OK upon success.
* Other error codes for failure.

### Example:

#include "EventHubClient.h"

#include "EventData.h"

int main(int argc, char\*\* argv)

{

    printf("Starting the EventHubClientSample...\n");

    const char\* connectionString = "Endpoint=sb://xxx.servicebus.windows.net/;SharedAccessKeyName=Device0SendAccessKey;SharedAccessKey=xyztdyfjhdhejwldif12bogus==";

    const char\* path = "tamer-eh-02";

    const char\* msgText = "Hello From EventHubClient";

    EVENTDATA\_HANDLE eventDataHandle = EventData\_CreateWithNewMemory(msgText, strlen(msgText));

    EVENTHUBCLIENT\_HANDLE clientHandle = EventHubClient\_CreateFromConnectionString(connectionString, path);

    if (clientHandle == NULL)

    {

        /\* error \*/

    }

    else

    {

        if (EventHubClient\_Send(clientHandle, eventDataHandle) != EVENTHUBCLIENT\_OK)

        {

            /\* error \*/

        }

    }

    EventData\_Destroy(eventDataHandle);

    EventHubClient\_Destroy(clientHandle);

    return 0;

}

## EVENTHUBCLIENT\_RESULT EventHubClient\_SendBatch(EVENTHUBCLIENT\_HANDLE clientHandle, EVENTDATA\_HANDLE\* eventDataHandle, size\_t count)

Synchronously sending an array of DataEvent of data blocks to an existing EventHub.

### Arguments:

* *clientHandle* – Handle created by a call to the EventHubClient\_CreateFromConnectionString.
* *eventDataHandle –* An array of EventData handles.
* *count* – The number of EventData blocks to be sent.

### Returns:

* EVENTHUBCLIENT\_OK upon success.
* Other error codes for failure.

### Example:

#include "EventHubClient.h"

#include "EventData.h"

int main(int argc, char\*\* argv)

{

    printf("Starting the EventHubClientSample...\n");

const char\* connectionString = "Endpoint=sb://xxx.servicebus.windows.net/;SharedAccessKeyName=Device0SendAccessKey;SharedAccessKey=xyztdyfjhdhejwldif12bogus==";

    const char\* path = "eventHubName";

    const char\* msgText0 = "Hello From EventHubClient0";

    const char\* msgText1 = "Hello From EventHubClient1";

EVENTDATA\_HANDLE eventDataHandles[2] = { EventData\_CreateWithNewMemory(msgText0, strlen(msgText0)),

EventData\_CreateWithNewMemory(msgText1, strlen(msgText1))

 };

    EVENTHUBCLIENT\_HANDLE clientHandle = EventHubClient\_CreateFromConnectionString(connectionString, path);

    if (clientHandle == NULL)

    {

        /\* error \*/

    }

    else

    {

        if (EventHubClient\_SendBatch(clientHandle, eventDataHandles, countOfData) != EVENTHUBCLIENT\_OK)

        {

            /\* error \*/

        }

    }

EventData\_Destroy(eventDataHandles[0]);

EventData\_Destroy(eventDataHandles[1]);

    EventHubClient\_Destroy(clientHandle);

    return 0;

}

## EVENTHUBCLIENT\_RESULT EventHubClient\_SendAsync(EVENTHUBCLIENT\_HANDLE clientHandle, EVENTDATA\_HANDLE eventDataHandle, EventHubClient\_CallBack eventHubClientCallBack, void\* callbackContext)

Asynchronously sends a single block of data to an existing EventHub. The consumer must provide a callback function that will be invoked upon receiving acknowledgment from EventHub that data has been received.

### Arguments:

* *clientHandle* – Handle created by a call to the EventHubClient\_CreateFromConnectionString.
* *eventDataHandle –* Handle to an EventData instance which encapsulates the data to be sent to EventHub.
* *eventHubClientCallBack* – A user defined function to be invoked.
* *callbackContext –* Context associated with the eventHubClientCallBack

### Returns:

* EVENTHUBCLIENT\_OK upon success.
* Other error codes for failure.

### Example:

#include <stdio.h>

#include "EventHubClient.h"

#include "EventData.h"

void CallMeBack(EVENTHUBCLIENT\_HANDLE clientHandle, void\* context)

{

printf("Sent ...\n");

}

int main(int argc, char\*\* argv)

{

printf("Starting the EventHubClientSample...\n");

const char\* connectionString = "Endpoint=sb://xxx.servicebus.windows.net/;SharedAccessKeyName=Device0SendAccessKey;SharedAccessKey=xyztdyfjhdhejwldif12bogus==";

const char\* path = "tamer-eh-02";

const char\* msgText = "Hello From EventHubClient";

EVENTDATA\_HANDLE eventDataHandle = EventData\_CreateWithNewMemory(msgText, strlen(msgText));

EVENTHUBCLIENT\_HANDLE clientHandle = EventHubClient\_CreateFromConnectionString(connectionString, path);

if (clientHandle == NULL)

{

/\* error \*/

}

else

{

/\* when sent call the CallMeBack function and pass the clientHandle as context \*/

if (EventHubClient\_SendAsync(clientHandle, eventDataHandle, CallMeBack, clientHandle) != EVENTHUBCLIENT\_OK)

{

/\* error \*/

}

else

{

getchar();

}

}

EventData\_Destroy(eventDataHandle);

EventHubClient\_Destroy(clientHandle);

return 0;

}

## EVENTHUBCLIENT\_RESULT EventHubClient\_SendBatchAsync(EVENTHUBCLIENT\_HANDLE clientHandle, EVENTDATA\_HANDLE\* eventDataHandle, size\_t count, EventHubClient\_CallBack eventHubClientCallBack, void\* callbackContext)

Asynchronously sends a set of data blocks to an existing EventHub. The consumer must provide a callback function that will be invoked upon receiving acknowledgment from EventHub that data has been received.

### Arguments:

* *clientHandle* – Handle created by a call to the EventHubClient\_CreateFromConnectionString.
* *eventDataHandle –* An array of EventData handles.
* *count* – The number of EventData blocks to be sent.
* *eventHubClientCallBack* – A user defined function to be invoked.
* *callbackContext –* Context associated with the eventHubClientCallBack

### Returns:

* EVENTHUBCLIENT\_OK upon all batched data has been sent successfully.
* Other error codes for failure.

### Example:

#include <stdio.h>

#include "EventHubClient.h"

#include "EventData.h"

void CallMeBack(EVENTHUBCLIENT\_HANDLE clientHandle, void\* context)

{

printf("Sent ...\n");

}

int main(int argc, char\*\* argv)

{

printf("Starting the EventHubClientSample...\n");

const char\* connectionString = "Endpoint=sb://xxx.servicebus.windows.net/;SharedAccessKeyName=Device0SendAccessKey;SharedAccessKey=xyztdyfjhdhejwldif12bogus==";

const char\* path = "eventHubName";

const char\* msgText0 = "Hello From EventHubClient0";

const char\* msgText1 = "Hello From EventHubClient1";

EVENTDATA\_HANDLE eventDataHandles[2] = { EventData\_CreateWithNewMemory(msgText0, strlen(msgText0)),

EventData\_CreateWithNewMemory(msgText1, strlen(msgText1))

};

EVENTHUBCLIENT\_HANDLE clientHandle = EventHubClient\_CreateFromConnectionString(connectionString, path);

if (clientHandle == NULL)

{

/\* error \*/

}

else

{

if (EventHubClient\_SendBatchAsync(clientHandle, eventDataHandles, countOfData, CallMeBack, clientHandle) != EVENTHUBCLIENT\_OK)

{

/\* error \*/

}

else

{

getchar();

}

}

EventData\_Destroy(eventDataHandles[0]);

EventData\_Destroy(eventDataHandles[1]);

EventHubClient\_Destroy(clientHandle);

return 0;

}