Publish and Subscribe with Broker ActiveX Control

Broker ActiveX Control provides five Broker functions to enable publishing and subscription. Publish and subscribe enables an application to send a message (publication) to multiple receivers (subscribers).

This functionality is supported by the native COM interface as well as by the Transaction Object Repository interface (TOR file).

Some examples of publish and subscribe for the native interface are given below.

This chapter covers the following topics:

- Writing Subscriber Applications
- Writing Publisher Applications

Writing Subscriber Applications

A subscriber receives the publications that are sent by the publisher. Subscribers will only receive publications that are sent *after* they have subscribed to a topic. Similarly, publishers can only send a publication if at least one subscriber has already subscribed to a topic.

To learn more about a particular topic, see *Writing Applications: Publish and Subscribe*.

The methods, functions, properties and steps required to operate as subscriber are described here.

Methods

Method	Description
`InvokeBrokerFunction`	Invoke the Broker function call.
`GetReceiveData`	Return the most recently received publication as string.

Functions

These will be set in the `function` property. For all function calls, the `UserID`, `Password` (if security Broker), `Token` and `Topic` properties must be set.

Function	Option
Logon	
Subscribe	Option = None
Receive Publication	Option = None, Publication ID = NEW
Control Publication	Option = Commit
Unsubscribe	Option = None
Logoff	

Properties

Property	Description
`APIVersion`	Must be set to 8 or above
`Function`	See the Functions table.
`Option`	Needed to receive and control publications.
`UserID`	Your user ID.
`Password`	Your password.
`Wait`	Set an adequate amount of time to wait for a publication. The length of time depends on your application and can be set to Yes to wait until a publication has been received.
`UOWstatus`	Broker returns the current status of the publication.
`ReceiveBufferLength`	Set to the maximum possible publication length.
`Token`	Additional caller identifier. The combination of the user ID and the token must be unique.
`Topic`	The topic of the publication that is to be received. Use a topic that has been registered with the EntireX Broker. Ask your broker administrator to get a valid topic.
`Publication ID`	Always NEW for the first call to receive a publication. For subsequent messages, reuse the received publication ID. See the step Check UOW status to find out whether it is a multi-message publication.

Important:
Please check the Error Status regularly; at least after every `InvokeBrokerFunction`.

To operate as subscriber

1. Set the `APIVersion` property to 8, the functionality Publish and Subscribe is only available with API version 8 and above.
2. Set the `BrokerID` property for your EntireX Broker.

3. Set the `UserID`, `Password` (if required), `Token` and `Topic` properties.
4. Set the `Option` property to 0 (None).
5. Set the `Function` property to 9 (Logon). You must be logged on to use the publish-and-subscribe functionality.
6. Call the method `InvokeBrokerFunction` to perform the logon function. The application has now been logged on to the EntireX Broker.
7. After successful logon to the Broker, set the `function` property to 19 (subscribe).
8. Call the method `InvokeBrokerFunction` to subscribe. The application has now been subscribed as a non-durable subscriber for this topic. If you want to be a durable subscriber, set the `Option` property to **Durable** when calling the method `InvokeBrokerFunction`. To learn more about the difference between durable and non-durable subscribers, see *Concepts of Persistent Messaging*.
9. Set the `Wait` property to the required value, for example 60s (s = seconds).
10. Set the `Option` property to 0 (None).
11. Set the `PublicationID` property to NEW.
12. Set the `Function` property to 18 (Receive Publication).
13. Set the `ReceiveBufferLength` property to your maximum expected publication length (can be up to 2048).
14. Call the method `InvokeBrokerFunction` to receive publications. The application will now wait to receive a publication. With the current settings the application would receive a publication within 60 seconds or time out after 60 seconds. If the publication is larger than 2048 characters, Broker ActiveX Control will return an error. Assuming that an application has received a publication, that publication now has a publication ID, assigned to it by the EntireX Broker.
15. Get the received data with the method `GetReceiveData`.
16. The current status of the publication is stored in the `UOWstatus` property. Check this UOWstatus now. A UOWstatus of 12 (Received Only), means that the received publication has only one message. A UOWstatus of 9 (Received First) means that you have received the first message of a multi-message publication. In this case you should request the other messages of this publication, until a UOWstatus of 11 (Received Last) is returned. See *Concepts of Persistent Messaging* for more information. To inform the EntireX Broker that the subscriber has received and retrieved the publication the subscriber must commit this.
17. Do not change the `PublicationID` property. This is required to refer to the received publication.
18. Set the `Option` property to 10 (Commit).
19. Set the `Function` property to 21 (**Control Publication**).
20. Call the method `InvokeBrokerFunction` to control the publication.

21. Get the `UOWstatus` property and check the status. The value of the UOWstatus should now be 5 (Processed). Your application may now run in a loop between steps 9 and 21 to receive several publications.

22. Set the `Option` property to 0 (None).

23. Set the `Function` property to 20 (Unsubscribe).

24. Call the method `InvokeBrokerFunction` to unsubscribe. The application has now been unsubscribed from the topic.

25. Set the `Function` property to 10 (Log off).

26. Call the method `InvokeBrokerFunction` to log off. The application has now been logged off from the EntireX Broker.

C# Example with a simple Subscriber who has Received only one Publication

```
using System;
// add the "EntireX Broker ActiveX Control" in COM references
using BrokerLib;

namespace Pubsub
{
    class Class1
    {
        static BrokerClass ebx;
        // EntireX Broker ACI definitions.
        const int function_logon = 9;
        const int function_logoff = 10;
        const int function_subscribe = 19;
        const int function_unsubscribe = 20;
        const int function_receive_publication = 18;
        const int function_control_publication = 21;
        const int option_none = 0;
        const int option_commit = 10;
        const int uowstatus_receive_only = 12;
        const int uowstatus_receive_last = 11;

        // procedure to invoke an entirex broker function call.
        static bool invokeEBX(short function, short option)
        {
            bool rc = true;
            ebx.Option = option;
            ebx.Function = function;
            ebx.InvokeBrokerFunction();
            // check the error status after the broker call.
            if (ebx.ErrorCode != "00000000")
            {
                Console.WriteLine(ebx.ErrorMsg);
                rc = false;
            }
            return rc;
        }

        [STAThread]
        static void Main(string[] args)
        {
            bool receive_error = false;
            bool subscribe_error = false;
```

```
            ebx = new BrokerClass();

            ebx.APIVersion = 8;
            ebx.BrokerID = "localhost";
            ebx.UserID = "EBXUSER";
            ebx.Token  = "EBXTOKEN";
            ebx.Topic  = "NYSE";

            Console.WriteLine("Log on");
            if (!invokeEBX(function_logon, option_none))
                return; // logon failed

            Console.WriteLine("Subscribe");
            if (!invokeEBX(function_subscribe, option_none))
                subscribe_error = true; // subscribe failed

            if (!subscribe_error)
            {
                ebx.PublicationID = "NEW";
                ebx.ReceiveBufferLength = 2048;
                ebx.Wait = "60s";
                // loop until all messages of the publication have been received.
                do
                {
                    Console.WriteLine("Receive Publication");
                    if (!invokeEBX(function_receive_publication, option_none))
                    {
                        receive_error = true;   // receive failed
                        break;          // cancel the while loop
                    }
                    else
                    {
                        // work with the received publication.
                        Console.WriteLine(ebx.GetReceiveData());
                    }
                } while ((ebx.UOWStatus != uowstatus_receive_only) &&
                       (ebx.UOWStatus != uowstatus_receive_last));

                if (!receive_error)
                {
                    Console.WriteLine("Control Publication");
                    invokeEBX(function_control_publication, option_commit);
                    // the publication status should be 5 (= processed)
                    Console.WriteLine("Publication status = " + ebx.UOWStatus);
                }
                Console.WriteLine("Unsubscribe");
                invokeEBX(function_unsubscribe, option_none);
            }
            Console.WriteLine("Log off");
            invokeEBX(function_logoff, option_none);
        }
    }
}
```

Writing Publisher Applications

The publisher sends publications to subscribers. Publications will fail if there is no subscriber for this topic. See *Writing Applications: Publish and Subscribe* for a list of the valid topics.

The methods, functions, properties and steps required to operate as Publisher are described below.

Methods

Method	Description
`InvokeBrokerFunction`	Invoke the broker function call.
`SetSendData` or `SetSendDataLong`	Set the publication to be sent.

Functions

These will be set in the `Function` property. For all function calls, the `UserID`, `Password` (if secure Broker), `Token` and `Topic` properties must be set.

Function	Option
Logon	
Send Publication	Option = Sync, Publication ID = NEW.
Control Publication	Option = Commit. A publication can also be committed with function=send_publication option=commit.
Logoff	

Properties

Property	Description
`APIVersion`	Must be set to 8 or above.
`Function`	See the function table.
`Option`	Needed to send and control publication.
`UserID`	Your user ID.
`Password`	Your password.
`Wait`	Must be set to NO.
`UOWstatus`	Broker returns the current status of the publication.
`Token`	Additional identifier of the caller. The combination of the user ID and the token must be unique.
`Topic`	The topic of the publication that is to be received. Use a topic that has been registered with the EntireX Broker. Ask your Broker administrator to get a valid topic.
`PublicationID`	Always NEW for the first call to send a publication. If you want to send a multi-message publication, reuse the received publication ID to send the other messages.

Important:
Please check the Error Status regularly; at least after every `InvokeBrokerFunction`.

To operate as publisher

1. Set the `APIVersion` property to 8, the publish-and-subscribe functionality is only available with API version 8 or above.
2. Set the `BrokerID` property for your EntireX Broker.
3. Set the `UserID`, `Password` (if required), `Token` and `Topic` properties.
4. Set the `Option` property to 0 (None).
5. Set the `Function` property to 9 (Logon). You must log on to use the publish-and-subscribe functionality.
6. Call the method `InvokeBrokerFunction` to perform the Logon function.

   The application has now been logged on to the EntireX Broker.
7. Set the `Option` property to 10 (Commit).
8. Set the `Function` property to 17 (Send Publication).
9. Set the `Wait` property to NO.
10. Set the `PublicationID` property to NEW.
11. Call the method `SetSendData` or `SetSendDataLong` to set the publication data.
12. Call the method `InvokeBrokerFunction` to send the publication.
13. Get the `UOWstatus` property and check this. It should be 2 (Accepted).

    A publication has now been sent. Please note that the publication will fail if there are no subscribers to this topic. If your publication has more than one message, the steps beginning with Set the `Option` property to 10 (Commit) will change. See *Concepts of Persistent Messaging*.
14. Set the `Option` property to 0 (None).
15. Set the `Function` property to 10 (Logoff).
16. Call the method `InvokeBrokerFunction` to log off.

    The application has now been logged off from the EntireX Broker.

C# Example with a simple Publisher who Sends only one (single-message) Publication

```
using System;
// add the "EntireX Broker ActiveX Control" in COM references
using BrokerLib;

namespace Pubsub
```

```
{
    class Class1
    {
        static BrokerClass ebx;
        // EntireX Broker ACI definitions
        const int function_logon = 9;
        const int function_logoff = 10;
        const int function_send_publication = 17;
        const int function_control_publication = 21;
        const int option_none = 0;
        const int option_commit = 10;

        // procedure to invoke an entirex broker function call
        static bool invokeEBX(short function, short option)
        {
            bool rc = true;
            ebx.Option = option;
            ebx.Function = function;
            ebx.InvokeBrokerFunction();
            if (ebx.ErrorCode != "00000000")
            {
                Console.WriteLine(ebx.ErrorMsg);
                rc = false;
            }
            return rc;
        }

        [STAThread]
        static void Main(string[] args)
        {
            ebx = new BrokerClass();
            String s = "A small c# publisher example with EntireX Broker ActiveX Control.";

            ebx.APIVersion = 8;
            ebx.BrokerID = "localhost";
            ebx.UserID = "EBXUSER";
            ebx.Token  = "EBXTOKEN";
            ebx.Topic  = "NYSE";

            Console.WriteLine("Log on");
            if (!invokeEBX(function_logon, option_none))
                return; // logon failed

            ebx.Wait = "NO";            // set to NO because we cannot receive data
            ebx.PublicationID = "NEW";
            ebx.SetSendDataLong(s, s.Length);   // set the sent data

            Console.WriteLine("Send Publication");
            invokeEBX(function_send_publication, option_commit);
            // Check the status of the UOW. It should be 2 (= Accepted).
            Console.WriteLine("Publication status = " + ebx.UOWStatus);

            Console.WriteLine("Log off");
            invokeEBX(function_logoff, option_none);
        }
    }
}
```