Gallagher Command Centre

OPC Integrator's Guide

Version: 3.00
Date: May 2011

Disclaimer

This document gives certain information about products and/or services provided by Gallagher Group Limited or its related companies (referred to as “Gallagher Group”).

The information is indicative only and is subject to change without notice meaning it may be out of date at any given time. Although every commercially reasonable effort has been taken to ensure the quality and accuracy of the information, Gallagher Group makes no representation as to its accuracy or completeness and it should not be relied on as such. To the extent permitted by law, all express or implied, or other representations or warranties in relation to the information are expressly excluded.

Neither Gallagher Group nor any of its directors, employees or other representatives shall be responsible for any loss that you may incur, either directly or indirectly, arising from any use or decisions based on the information provided.

Table of Contents

Introduction

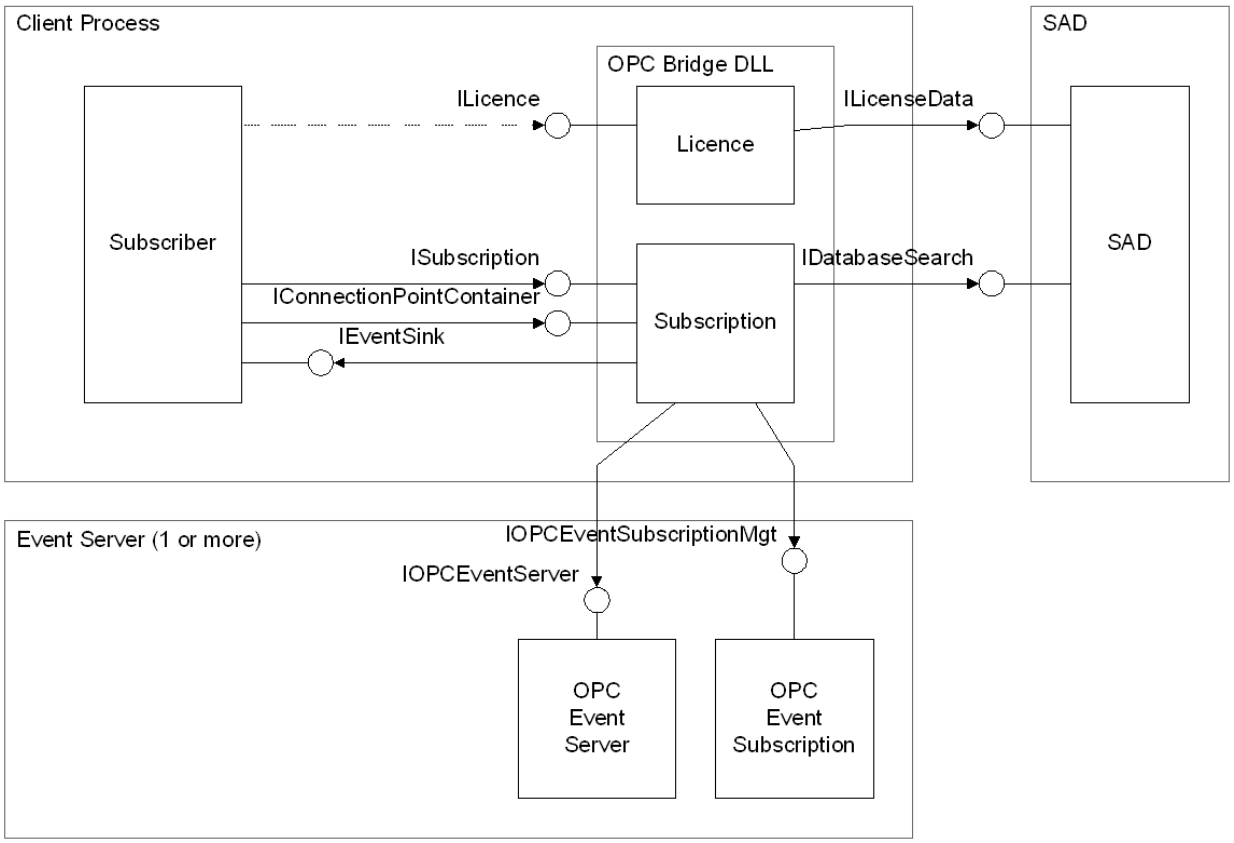
This guide is intended for developers using the OPC Bridge to work with Gallagher Command Centre and its software to monitor events and alarms in the Gallagher Command Centre system.

Basic Architecture

The OPC Bridge works as a simplified lightweight interface for working with the Gallagher Command Centre systems' more complicated event architecture. While the bridge abstracts out much of the detail of using the OPC Interfaces that Gallagher Command Centre supports, most of the functionality has remained intact.

The OPC Bridge has two main components:

- the Licence component, and
- the Subscription Component.



Licence Component

The licence component is used to determine whether a particular feature of Gallagher Command Centre is available. The Licence component will interrogate the SAD (Server Access to Data) to determine whether a particular

feature is available. This data is for the purposes of the Client process to determine if it is licensed to run, and does not in fact participate in any way with the OPC Interfaces published by the Gallagher Command Centre software.

Subscription Component

This component is the main component of the OPC Bridge. The Subscription Component is responsible for fetching the locations of the event servers from the SAD, and connecting to those event servers. The Subscription Component monitors the events, and can disconnect and reconnect to the event servers as is required.

To receive notifications of events and alarms, the Subscriber in the client process registers its own IEventSink with the subscription object using the Subscription Component's IConnectionPointContainer Interface.

Using the OPC Bridge with Visual Basic

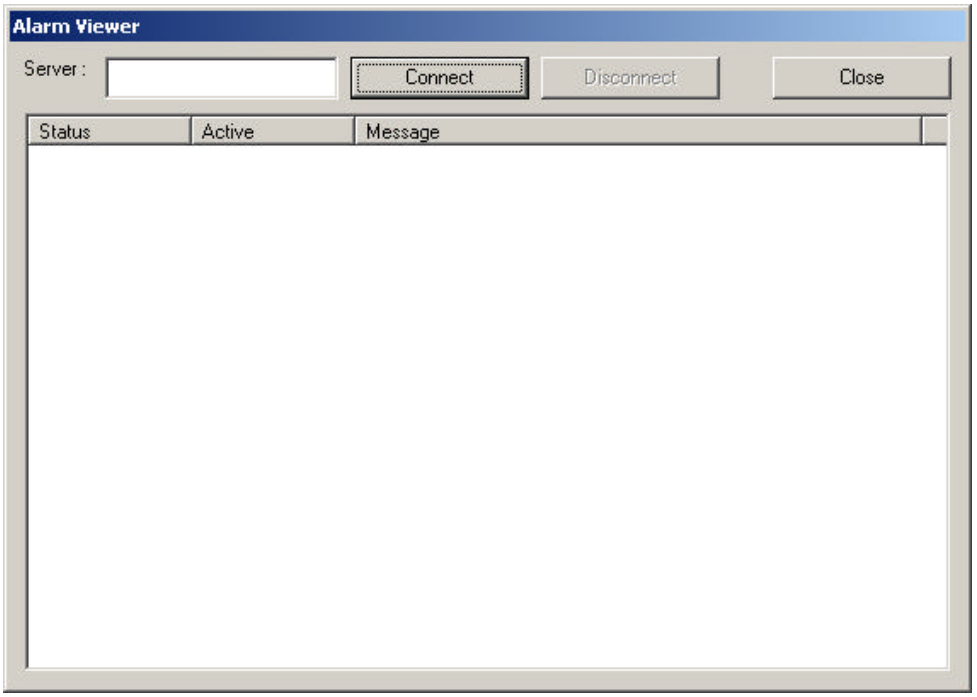
Introduction

This Tutorial demonstrates the implementation of a simple Alarm Viewer in Visual Basic. Using this alarm viewer it will be possible to see the state of all alarms which are unprocessed and are available through the OPC Bridge.

Screen Layout

The following controls will be used in this tutorial:

- ServerName (Text Box) for setting the name of the server.
- ConnectButton (Button) for connecting to the event server.
- DisconnectButton (Button) for disconnecting from the event server.
- CloseButton (Button) for quitting the program.
- AlarmWindow (ListView from Microsoft Window Common Controls 6.0) for showing the alarms.
  It is also necessary to add a reference to the OPCBridge v1.0 type library in the project references so that it's possible to use the OPC Bridge Library.



General Structure

Let's start with a general architecture. The following code should be added to the main form's code:

```
Dim WithEvents oSubscription As OPCBRIDGELib.Subscription

Private Sub CloseButton_Click()
    'exit the program
    End
End Sub

Private Sub ConnectButton_Click()
    'connecting disables connect button and the Server Name textbox
    ConnectButton.Enabled = False
    ServerName.Enabled = False

    '...

    'connect code goes here

    '...

    'at end of function enable disconnect button
    DisconnectButton.Enabled = True
End Sub

Private Sub DisconnectButton_Click()
    'disconnecting disables the disconnect button
    DisconnectButton.Enabled = False

    '...

    'disconnect code goes here

    '...

    'at the end of the function enable the connect button
    ConnectButton.Enabled = True
    'and the Server Name textbox
    ServerName.Enabled = True
End Sub

Private Sub Form_Load()
    'Disable the Disconnect Button
    DisconnectButton.Enabled = False
```

```
    'create the subscription
    Set oSubscription = New Subscription
End Sub


Private Sub oSubscription_OnEvent(ByVal pNotification As _
OPCBRIDGELib.IEventNotification)
    'Event Handler here


End Sub
```

At this point it should be possible to test the program, click a few buttons, and close the program.

Setting the Filter

To receive events from the OPC Bridge and the event server, it is necessary to tell the bridge which kind of events you want to be sent. In the case of this example, we are going to enable all of the event group types. Add the following connection code to the ConnectButton_Click subroutine

```
'set the server name
    oSubscription.Server = ServerName.Text

    'set the types
    Dim oAvailableGroups As OPCBRIDGELib.EventGroups
    Dim oGroups As OPCBRIDGELib.EventGroups
    Dim oGroup As OPCBRIDGELib.EventGroup

    'get the available groups
    Set oAvailableGroups = oSubscription.GetAvailableGroups

    'and a reference to the groups
    Set oGroups = oSubscription.Groups

    'add the groups to the subscription
    For Each oGroup In oAvailableGroups
        oGroups.Add oGroup
    Next oGroup

    'set up the rest of the filter

    'min and max priority
    oSubscription.MaximumPriority = 9
    oSubscription.MinimumPriority = 1
```

```
'check the connection every 10 seconds
oSubscription.CheckEnabled = 1 'true'
oSubscription.CheckInterval = 10

'attempt to connect once every second
oSubscription.ConnectAttempts = 1
oSubscription.ConnectInterval = 1

'don't notify me if it takes a little while to connect
oSubscription.NotifyOnFailure = 0 'false'

'ACTIVATE IT!!
oSubscription.Activate

'ask for the latest news
oSubscription.Refresh
```

At this point your Alarm viewer will be receiving events, it just won't be doing anything with them.

Handling the Events

Since this is an alarm viewer, we only want to show events that are *alarms* in the alarm window. It is easy to determine which event is an alarm as the Notification Objects have an alarm property, which is set to 1(true) if it is an alarm notification and 0(false) if it is not. To distinguish between two alarms, we use the Cookie property of the event, which is what we are going to use as a Tag in the AlarmWindow listview.

Add the following code to the oSubscription_OnEvent subroutine

```
If pNotification.Alarm = 0 Then
    Exit Sub
End If

'stringify the cookie
Dim StrCookie As String
StrCookie = Str(pNotification.Cookie)

'try to find the cookie in the listview
Dim CurrItem As ListItem
Dim FoundItem As ListItem
Dim FoundAnItem As Boolean
FoundAnItem = False

For Each CurrItem In AlarmWindow.ListItems
```

```
    If CurrItem.Tag = StrCookie Then
        Set FoundItem = CurrItem
        FoundAnItem = True
    End If
Next

'if there wasn't one, create one
If FoundAnItem = False Then
    Set FoundItem = AlarmWindow.ListItems.Add
    FoundItem.Tag = StrCookie
End If

If pNotification.Processed = 1 Then
    'remove the item as it is no longer an alarm
    AlarmWindow.ListItems.Remove FoundItem.Index
    Exit Sub
ElseIf pNotification.Acknowledged = 1 Then
    'it is acknownedged
    FoundItem.Text = "Acked"
Else
    'it isn't acknowledged
    FoundItem.Text = "Not Acked"
End If

'test whether the alarm is active
If pNotification.Active = 1 Then
    FoundItem.SubItems(1) = "Active"
Else
    FoundItem.SubItems(1) = "Inactive"
End If

'add the message
FoundItem.SubItems(2) = pNotification.Message
```

Disconnecting

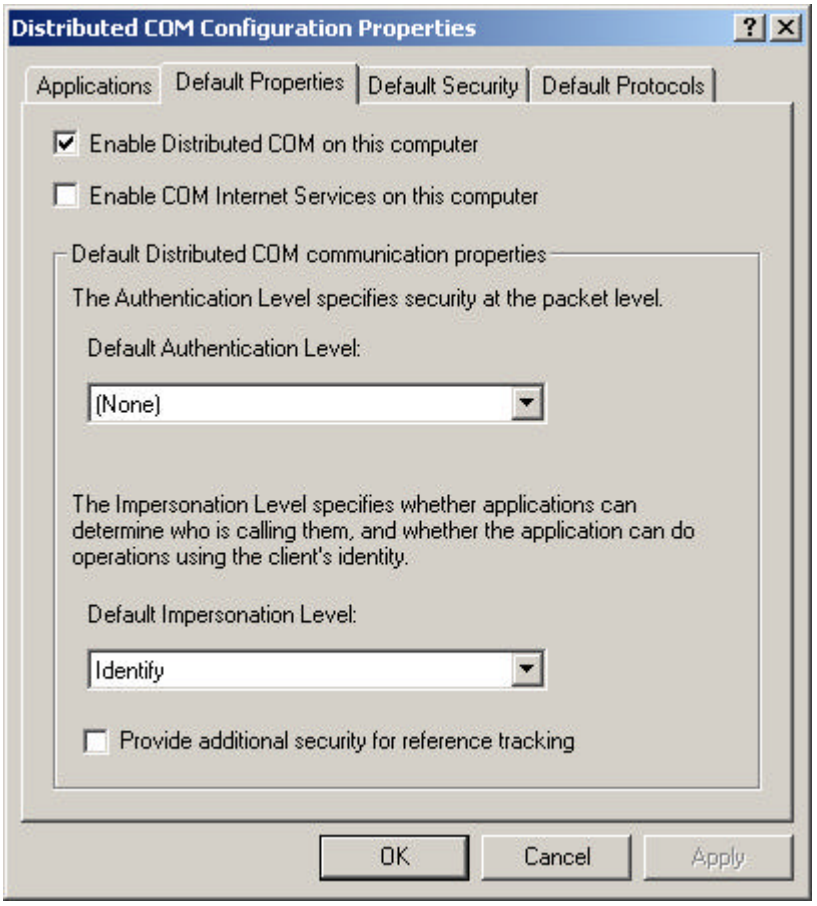
Disconnecting from the event server is fairly straightforward, compared to the other steps. To deactivate any subscriptions that the OPC Bridge has with any event servers, simply call the subscription component's Deactivate member.

```
'deactivate the subscription
oSubscription.Deactivate

'clear the alarm window
AlarmWindow.ListItems.Clear
```

Security Issues

If the application is to be used between machines on seperate domains or to connect to a server on a workgroup, then it is necessary to set the security of the process. This is done with a call to the windows API function CoInitializeSecurity with authentication level none and impersonation level identify. If the programming environment does not allow for the process to call this function then setting the defaults with dcomcnfg is preferable. In this case it is also necessary to add a user to the default access permissions on the default security tab so that the bridge can log into the remote machine.



Other Issues

If this program was going to be used in a commercial setting there would need to be some kind of error handling for the cases when the program's input is not exactly what it is expected (for example the event server doesn't exist). Also to run with Gallagher Command Centre the OPC bridge must have a way to log on. This will be an operator with username "opc" and password "" (blank).

Quick Reference

Interfaces

ICustomPropertyBag

Contains properties from the EventNotification component.

Methods

Read	Returns a given property.

IEventGroup

Properties

Description	Description of the event type
ID	Category number of the event type

IEventGroups

Properties

_NewEnum	getting this property retrieves a new enumerator of type IEnumVARIANT
Count	number of items contained
Item	get a particular item from the IEventGroups Container (item with a particular group ID)

Methods

Add	Add a group to the collection
Remove	remove a group from the collection indexed by it's ID

IEventNotification

Contains information about an event.

Properties

Acknowledged	1 if the event is an alarm that has been acknowledged, 0 otherwise
Active	1 if the event is an alarm that is active, 0 otherwise
Alarm	1 if the event is an alarm, 0 otherwise
ArrivalTime	Time that the event arrived at the event server
Cookie	Unique identifier if the event is an alarm, 0 otherwise
Description	Description of the event

Details	Detail string for the event
Group	The category into which the event falls
History	A multi-line string detailing the history of the alarm
Message	The message field displayed in the event window
Priority	The priority of the message from 0 to 9
Processed	1 if the event is an alarm which has been processed, 0 otherwise
Properties	A CustomPropertyBag containing all of the properties for the Event
Refresh	1 if the event is in response to an alarm refresh (call to ISubscription::Refresh), 0 otherwise
Source	the source of the event
Time	the time that the event occurred

IEventSink

This is the interface that the Subscription component uses to notify the subscriber of events and failures of the Subscription component to connect.

Methods

OnEvent	Method called when an event arrives at the OPC Bridge
OnFailure	Method called when the Subscription Component fails to connect to an event server.

ILicence

This interface is used to determine whether a given feature is available/licensed or to retrieve values from the Gallagher Command Centre licence file.

Properties

Server	The name of the server to query for the available features

Methods

IsFeatureAvailable	Method called to determine whether a given feature is available
GetValue	Method called to read a value from the licence file

ISubscription

Interface of the main component of the OPC Bridge. The subscription component monitors subscriptions with the event servers, handles connection and disconnection from the Event servers, and the setting and removing of filters.

Properties

AttemptsBeforeNotify	Connection attempts before calling the IEventSink::OnFailure method
CheckEnabled	1 to check the connection every CheckInterval Seconds
CheckInterval	the number of seconds to wait between checks to determine that the subscription is still active
ConnectAttempts	Number of attempts to connect to try every ConnectInterval Seconds
Enabled	Will be 1 if the subscription is active, 0 otherwise
Groups	Collection of IEventGroups interfaces to filter on
MaximumPriority	The maximum priority event to let through the filter
MinimumPriority	The minimum priority Event to let through the filter
NotifyOnFailure	1 if the Subscription is to IEventSink::OnFailure method when connection fails AttemptsBeforeNotify times, 0 otherwise
Server	The name of the server to which the Subscription will connect
UTCTime	Whether the OPC Bridge will send the Time and ArrivalTime fields in UTC Time.

Methods

Activate	Activate the subscription object, using the current filters
Deactivate	Deactivate the subscription object
GetAvailableGroups	Retrieve a list of available event categories
LoadConfiguration	Loads a previously stored configuration from an ini file
Refresh	requests the current alarms to refresh
SaveConfiguration	Stores the current configuration in an ini file

Methods

ICustomPropertyBag::Read

Read a property from the CustomPropertyBag.

Syntax

[Visual Basic] Read(pszPropName As String) As Variant
[C++\COM] HRESULT Read([in] LPCOLESTR pszPropName, [out, retval] VARIANT*pVal);

Parameters

pszPropName	The name of the property, can be one of: • Cardholder • EntryZone • CardNumber • FacilityCode • IssueLevel • Operator
pVal	a pointer to where the returned variant value of the property will be placed

Returns

S_OK	Call completed successfully.
E_PENDING	Another call to the Subscription object or one of its children is currently executing

Remarks

If there is no property then the return value will be of type VT_EMPTY.

Examples

[Visual Basic]

```
Public function ReadCardHolder(oPropertyBag As CustomPropertyBag) as
String
      ReadCardHolder = oPropertyBag.Read("Cardholder")
End function
```

[C++\Com]

```
BSTR ReadCardHolder(ICustomPropertyBag *pPropertyBag)
{
      BSTR bstrRetval;
      CComVariant V;
      pPropertyBag->Read(_T("Cardholder),&V);
      if(V.vt = VT_BSTR)
      {
            bstrRetval = SysAllocString(V.bstrVal);
      }
      else
      {
```

```
            bstrRetval = NULL;
        }
        return bstrRetval;
    }
```

IEventGroups::Add

Add an Event Group to the Collection.

Syntax

[Visual Basic] Add(pGroup As EventGroup)
[C++\Com] HRESULT Add([in] IEventGroup*pGroup);

Parameters

pGroup	Group to be added to the EventGroups collection.

Returns

S_OK	Group was added to the EventGroups collection
E_FAIL	Collection is currently read only
E_PENDING	Another call to the Subscription object or one of its children is currently executing

Remarks

If the IEventGroups is the Groups property from the ISubscription Interface, then any attempt to modify the groups while the subscription is active will fail. If the IEventGroups is the return value from ISubscription::GetAvailableGroups, then all modifications to the collection will fail.

Examples

See ISubscription::GetAvailableGroups for example of use.

IEventGroups::Remove

Remove an Event group from the collection

Syntax

[Visual Basic] Remove(nIndex As Long)
[C++\COM] HRESULT Remove([in] long nIndex);

Parameters

nIndex	The ID of the Group to remove

Returns

S_OK	Group was removed from the EventGroups collection
E_FAIL	Collection is currently read only
E_PENDING	Another call to the Subscription object or one of its children is currently executing

Examples

[Visual Basic]

```
Dim oGroup As EventGroup
Dim oGroups As EventGroups
Set oGroups = oSubscription.Groups
For Each oGroup In oGroups
      oGroups.Remove(oGroup.ID)
Next oGroup
```

[C++\COM]

```
HRESULT RemoveGroup(IEventGroup *pGroup, IEventGroups *pGroups)
{
      HRESULT hr = S_OK;
      try
      {
              long nID;
              hr = pGroup->get_ID(&nID);
              if(FAILED(hr))
              {
                      _com_issue_error(hr);
      }
      hr = pGroups->Remove(nID);
      if(FAILED(hr))
      {
      _com_issue_error(hr);
      }
      }
      catch(_com_error_e)
      {
      ATLTRACE(L"Error Occurred: %s\n", e.ErrorMessage());
      hr = e.Error();
      }
      return hr;
}
```

IEventSink::OnEvent

Method called when an event gets fired

Syntax

[Visual Basic] Sub OnEvent(ByVal pNotification As OPCBRIDGELib.IEventNotification)

[C++\COM] HRESULT OnEvent([in] IEventNotification *pNotification);

Parameters

pNotification	a pointer to an IEventNotification which contains the event notification information

Returns

The OPC Bridge ignores any return values

Examples

[Visual Basic]

```
Private Sub oSubscription_OnEvent(ByVal pNotification As _
OPCBRIDGELib.IEventNotification)
MsgBox pNotification.Message
End Sub
```

[C++\COM]

```
HRESULT CEventReceiver::OnEvent(IEventNotification*pNotification)
{
      BSTR bstrMessage;
      HRESULT hr;
      hr = pNotification->get_Message(&bstrMessage);
      if(FAILED(hr))
      {
            return hr;
      }
      MessageBox(GetFocus(), bstrMessage, NULL, MB_OK);
      SysFreeString(bstrMessage);
      return S_OK;
}
```

IEventSink::OnFailure

This is the method of the subscribing object to the OPC Bridge that gets called when an EventServer fails to `connect`.

Syntax

[Visual Basic] Sub OnFailure(ByVal failedType As Long, ByVal serverFailed As String)

[C++\COM] HRESULT OnFailure([in] long failedType,[in] BSTR serverFailed);

Parameters

failedType	The type of failure notification.
serverFailed	The name of the server that failed or was restored.

Returns

The OPC Bridge ignores any return value.

Remarks

If the Subscription fails to connect to any event Server ISubscription::AttemptsBeforeNotify times and ISubscription::NotifyOnFailure is set to 1, then the OnFailure method of the IEventSink will be called with failedType value bftServerFailed. If all of the servers have failed at this point, then the OnFailure method will be called again with bftAllServersFailed. When

any of the failed servers connects successfully, the OnFailure Method is called again with failedType bftServerFailed.
If failedType is bftServerFailed or bftAllServersFailed then the serverFailed parameter is the name of the server which failed or was restored, otherwise the serverFailed parameter will be NULL.

If the ISubscription::AttemptsBeforeNotify property is set to 0 then extra information about the nature of the failure will be presented.
The following values are valid for the failedType parameter:

```
bftServerFailed = 1,
bftServerRestored = 2,
bftAllServersFailed = 3,
bftDisconnectFailed = 10,
bftCouldNotCreateEventServer = 11,
bftCouldNotCreateEventSubscription = 12,
bftConnectionFailed = 13,
bftFilterProblem = 14,
bftBadConnectionPoint = 15,
bftConnectionCheckFailed = 16,
bftLoginFailedAuthentication = 20,
bftLoginFailedWorkstationLimit = 21,
bftLoginFailedAlreadyLoggedIn = 22,
bftLoginFailedLockOutUser = 23,
bftCatastrophicFailure = 50
```

Examples

[Visual Basic]

```
Private Sub oSubscription_OnFailure(ByVal failedType As Long , _
ByVal serverFailed as String)
      If failedType = bftServerFailed then
            MsgBox "Server " & serverFailed & " Failed"
      ElseIf failedType = bftServerRestored then
            MsgBox "Server " & serverFailed & " was restored"
      Else
            MsgBox "all servers failed"
      End If
End Sub
```

[C++\COM]

```
HRESULT CEventReceiver::OnFailure(long failedType, BSTR serverFailed)
{
      WCHAR wszOutString[1024];
      switch(failedType)
      {
            case bftServerFailed:
```

```
                {
                        wsprintf(wszOutString,L"Server %s
Failed",serverFailed);
                        MessageBox(GetFocus(), wszOutString, NULL,
MB_OK);
        }
                break;
                case bftServerRestored:
                {
                        wsprintf(wszOutString,L"Server %s
Restored",serverFailed);
                        MessageBox(GetFocus(), wszOutString, NULL,
MB_OK);
        }
                break;
                case bftAllServersFailed:
                {
                        MessageBox(GetFocus(), L"All Servers Failed",
NULL, MB_OK);
        }
                break;
        }
        return S_OK;
}
```

ILicence::IsFeatureAvailable

Tests whether a given feature is licensed.

Syntax

[Visual Basic] IsFeatureAvailable(bstrFeature As String) As long
[C++\Com] HRESULT IsFeatureAvailable([in] BSTR bstrFeature, [out, retval] BOOL *bRetval);

Parameters

bstrFeature	The name of the feature.
bRetval	A pointer to where the return value should be placed upon successful completion.

Returns

S_OK	Successful Completion.
E_PENDING	Another call to the Licence object is currently executing.
REGDB_E_CLASSNOTREG	one of the proxy Dlls may not be registered, or the Server field in the Licence object is not correctly set

Remarks

If the call succeeds then the variable pointed to by bRetval will be filled with either 1 if the feature is available or 0 if the feature is not available.

Examples

[Visual Basic]

```
Private Sub TestButton_Click()
    oLicence.Server = "CARDAX0"
    Dim oLicence As New OPCBRIDGELib.Licence
    If oLicence.IsFeatureAvailable(FeatureName.Text) = 1 Then
        MsgBox "feature Licensed"
    Else
        MsgBox "feature Unlicenced"
    End If
End Sub
```

[C++\Com]

```
bool TestPhotoID(ILicence *pLicence)
{
      BOOL FeatureAvailable;
      HRESULT hr = pLicence->put_Server(L"CARDAX0");
      if(FAILED(hr))
      {
             return false;
      }

     hr =pLicence->IsFeatureAvailable(L"Photo-ID",&FeatureAvailable);
      if(FAILED(hr))
      {
             return false;
      }
      return (FeatureAvailable == TRUE);
}
```

ILicence::GetValue

Retrieves a value from the licence file.

Syntax

[Visual Basic] GetValue(bstrSection As String, bstrKey As String, bstrDefault As String) As String

[C++\COM] HRESULT IsFeatureAvailable([in] BSTR bstrSection, [in] BSTR bstrKey, [in] BSTR bstrDefault, [out, retval] BSTR *bstrRetval);

Parameters

bstrSection	The name of the licence file section, (e.g. "Site" or "Limits")
bstrKey	The name of the licence file key, (e.g."Expiry" or "Cardholders")
bstrDefault	A value to return if the entry was not found in the licence file.
bstrRetval	A pointer to where the return value should be placed upon successful completion.

Returns

S_OK	Successful Completion.
E_PENDING	Another call to the Licence object is currently executing
REGDB_E_CLASSNOTREG	one of the proxy Dlls may not be registered, or the Server field in the Licence object is not correctly set

Remarks

If the call succeeds then the variable pointed to by bstrRetval will be filled with the value from the licence file or bstrDefault if the value was not found in the licence file.

Examples

[Visual Basic]

```
Private Sub TestButton_Click()
      oLicence.Server = "CARDAX0"
      Dim oLicence As New OPCBRIDGELib.Licence
      MsgBox oLicence.GetValue("Site", "Name", "Site Name Not Found")
End Sub
```

[C++\COM]

```
bool DisplaySiteName(ILicence *pLicence)
{
      HRESULT hr = pLicence->put_Server(L"CARDAX0");
      if(FAILED(hr))
      {
            return false;
      }

      CComBSTR SiteName;
      hr = pLicence->GetValue(L"Site", L"Name", L"Site Name Not
Found", &SiteName);
      if(FAILED(hr))
      {
            return false;
      }

      MessageBox(NULL, SiteName, L"Site Name", MB_OK);
      return true
}
```

ISubscription::Activate

Connects to any event servers.

Syntax

[Visual Basic] Activate
[C++\COM] HRESULT Activate();

Returns

S_OK	Successful Completion.
E_PENDING	Another call to the Subscription object or one of its children is currently executing
REGDB_E_CLASSNOTREG	one of the proxy Dlls may not be registered, or the Server field in the Subscription object is not correctly set

Remarks

If the Subscription is already active, then Activate will return S_OK without doing anything, however if the Subscription is not already active then the Subscription will create a monitor thread for each registered event server, and return. These worker threads will then attempt to connect to the event servers. There may be an unknown delay before the threads connect to any event servers, however if the event servers take more than AttemptsBeforeNotify tries to connect and NotifyOnFailure is 1 then a notification will be sent back to the subscriber via the IEventSink::OnFailure Member function.

Examples

[Visual Basic]

```
oSubscription.Activate
```

[C++\COM]

```
HRESULT hr;

hr = pSubscription->Activate();

if(FAILED(hr))

{

      MessageBox(GetFocus(),L"there was an error during
activation",NULL,MB_OK);
}
```

ISubscription::Deactivate

Deactivates the connection to any event servers that may be connected.

Syntax

[Visual Basic] Deactivate
[C++\COM] HRESULT Deactivate();

Returns

S_OK	Successful Completion
E_PENDING	Another call to the Subscription object or one of it's children is currently executing.
REGDB_E_CLASSNOTREG	one of the proxy Dlls may not be registered, or the Server field of the Subscription object is not correctly set

Remarks

The Deactivate member function requests the monitor threads to stop, and waits for them to do so.

Examples

[Visual Basic]

```
oSubscription.Deactivate
```

[C++\COM]

```
HRESULT hr;
hr = pSubscription->Activate();
if(FAILED(hr))
{
     MessageBox(GetFocus(),L"there was an error during
deactivation",NULL,MB_OK);
}
```

ISubscription::GetAvailableGroups

Performs a query on the database to determine the available event types.

Syntax

[Visual Basic] GetAvailableGroups() As EventGroups
[C++\COM] HRESULT GetAvailableGroups([out, retval] IEventGroups **ppGroups);

Parameters

ppGroups	The location of the Interface pointer to place the returned Collection into.

Returns

S_OK	Successful Completion
E_PENDING	Another call to the Subscription object or one of it's children is currently executing.
REGDB_E_CLASSNOTREG	one of the proxy Dlls may not be registered, or the Server field of the Subscription object is not correctly set

Remarks

The groups collection returned from GetAvailableGroups is read-only, and contains all of the groups that are recognised categories for the OPC Bridge.

Example

[Visual Basic]

```
Dim oAvailableGroups As OPCBRIDGELib.EventGroups
Dim oGroups As OPCBRIDGELib.EventGroups
Dim oGroup As OPCBRIDGELib.EventGroup

'get the available groups
Set oAvailableGroups = oSubscription.GetAvailableGroups

'and a reference to the groups
Set oGroups = oSubscription.Groups
```

```
'add the groups to the subscription
For Each oGroup In oAvailableGroups
      oGroups.Add oGroup
Next oGroup
```

[C++\COM]

```
CComPtr<IEventGroups> spEventGroups;
CComPtr<IEventGroups> spGroups;

//get the available groups
hr = m_spSubscription->GetAvailableGroups(&spEventGroups);
ATLASSERT(!FAILED(hr));

hr = m_spSubscription->get_Groups(&spGroups);
ATLASSERT(!FAILED(hr));

CComPtr<IUnknown> spUnk;
hr = spEventGroups->get__NewEnum(&spUnk);
ATLASSERT(!FAILED(hr));

CComPtr<IEnumVARIANT> spEnumVARIANT;
hr = spUnk.QueryInterface(&spEnumVARIANT);
ATLASSERT(!FAILED(hr));

for(;;){
      CComPtr<IEventGroup> spEventGroup;
      CComPtr<IDispatch> spDisp;
      CComVariant v;
      DWORD nFetched=0;

      hr = spEnumVARIANT->Next(1,&v,&nFetched);
      if(FAILED(hr)){
            break;
      }
      if(nFetched<1){
            break;
      }

      ATLASSERT(nFetched == 1);
      ATLASSERT(v.vt == VT_DISPATCH);

      spDisp = v.pdispVal;

      hr = spDisp.QueryInterface(&spEventGroup);
      ATLASSERT(!FAILED(hr));
      CComVariant v2(static_cast<IDispatch*>(spEventGroup));
```

```
        hr = spGroups->Add(v2);
        ATLASSERT(!FAILED(hr));
    }
```

ISubscription::LoadConfiguration

Loads a subscription configuration from a specified file and section.

Syntax

[Visual Basic] LoadConfiguration(filename As String, section As String)
[C++\COM] HRESULT LoadConfiguration([in] BSTR filename,[in] BSTR section);

Parameters

filename	the name of the file to load the configuration from
section	the name of the section of the file to load the configuration from

Returns

S_OK	Successful Completion
E_PENDING	Another call to the Subscription object or one of it's children is currently executing.
E_FAIL	The Subscription is currently Active
REGDB_E_CLASSNOTREG	one of the proxy Dlls may not be registered, or the Server field of the Subscription object is not correctly set

Remarks

The LoadConfiguration reads the configuration from a specified file, and also makes an internal call to ISubscription::GetAvailableGroups to correctly populate the Groups property of the Subscription component.
The other fields that are populated include ConnectAttempts, ConnectInterval, CheckEnabled, CheckInterval, MinimumPriority, and MaximumPriority. All other properties should remain the same. If the subscription is active then all attempts to modify the configuration will fail.

Examples

[Visual Basic]

```
oSubscription.Server = "CARDAX1"  'set a valid server

'load the previously saved configuration
oSubscription.LoadConfiguration "C:\MyConfig.ini", "Subscription"
```

[C++\COM]

```
HRESULT UpdateConfiguration(ISubscription*pSubscription)
{
    return pSubscription-
>LoadConfiguration(L"C:\\MyConfig.ini","Subscription");
}
```

ISubscription::Refresh

This member function will cause all of the alarms currently in the alarm stacks of all currently connected event servers that match the current filter to be retransmitted.

Syntax

[Visual Basic] Refresh
[C++\COM] HRESULT Refresh();

Returns

S_OK	Successful Completion

Remarks

This function will succeed whether or not the Subscription is enabled as when the subscription is not enabled there will be no event servers to update.

Examples

See Tutorial Step "Setting the filter".

ISubscription::SaveConfiguration

This member function saves the current configuration.

Syntax

[Visual Basic] SaveConfiguration(filename As String,section As String)
[C++\COM] HRESULT SaveConfiguration(BSTR filename, BSTR section);

Parameters

filename	the name of the file to save the configuration to
section	the section in the file to save the configuration to

Returns

S_OK	Successful Completion
E_PENDING	Another call to the Subscription object or one of it's children is currently executing.

Remarks

When Successful the SaveConfiguration Member function will save the configuration to the specified file and section.

Examples

[Visual Basic]

```
oSubscription.Server = "CARDAX1"  'set a valid server

'load the previously saved configuration
oSubscription.LoadConfiguration "C:\MyConfig.ini", "Subscription"
```

[C++\COM]

```
HRESULT UpdateConfiguration(ISubscription*pSubscription)
{
      return pSubscription-
>LoadConfiguration(L"C:\\MyConfig.ini","Subscription");
}
```

Properties

IEventGroup::Description

The description of the event group.

Syntax

[Visual Basic] Description As String
[C++\COM] HRESULT get_Description([out, retval] BSTR*pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only.

IEventGroup::ID

The ID of the Category the event group represents.

Syntax

[Visual Basic] ID As Long
[C++\COM] HRESULT get_ID([out,retval] Long *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only.

IEventGroups::_NewEnum

Retrieves a new enumerator for the EventGroups

Syntax

[Visual Basic] cannot be explicitly used in Visual Basic
[C++\COM] HRESULT get__NewEnum([out,retval] IEnumVariant **ppVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only. This function is called by Visual Basic when a programmer iterates over an EventGroups Collection using the For Each ... in ... statement.

Examples

For an example of the use of this property see ISubscription::GetAvailableGroups

IEventGroups::Count

Returns the number of items in the EventGroups object

Syntax

[Visual Basic] Count as Long
[C++\COM] HRESULT get_Count([out,retval] long *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only.

IEventGroups::Item

Returns a given item in the EventGroups object.

Syntax

[Visual Basic] Item(Index as Long) As EventGroup
[C++\COM] HRESULT get_Item([in] long Index,[out,retval] IEventGroup **ppVal);

Returns

S_OK	Successful Completion
E_FAIL	There is no Item with this index.
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only.

IEventNotification::Acknowledged

Whether an alarm has been acknowledged

Syntax

[Visual Basic] Acknowledged As Long
[C++\COM] HRESULT get_Acknowledged([out,retval] BOOL *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only. If the value of this property is 1 then the Event Notification represents an alarm that has been acknowledged, otherwise the value of this property will be 0.

IEventNotification::Active

Whether an alarm is currently active

Syntax

[Visual Basic] Active As Long
[C++\COM] HRESULT get_Active([out,retval] BOOL *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only. If the value of this property is 1 then the Event Notification represents an alarm that is currently active, otherwise the value of this property will be 0.

IEventNotification::Alarm

Whether the event is an alarm

Syntax

[Visual Basic] Alarm As Long
[C++\COM] HRESULT get_Alarm([out,retval] BOOL *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only. If the value of this property is 1 then the Event Notification represents an alarm, otherwise the value of this property will be 0.

IEventNotification::ArrivalTime

When the event arrived at the Event Server.

Syntax

[Visual Basic] ArrivalTime As Date
[C++\COM] HRESULT get_ArrivalTime([out,retval] DATE *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only.

IEventNotification::Cookie

A unique number for an alarm.

Syntax

[Visual Basic] Cookie As Long
[C++\COM] HRESULT get_Cookie([out,retval] long *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only. If the event is an alarm then the cookie will be a unique identifier, otherwise it will be 0.

IEventNotification::Description

A description of the event.

Syntax

[Visual Basic] Description As String
[C++\COM] HRESULT get_Description([out,retval] BSTR *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only.

IEventNotification::Details

The Details String for the Event Notification.

Syntax

[Visual Basic] Details As String
[C++\COM] HRESULT get_Details([out,retval] BSTR *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only.

IEventNotification::Group

The Group that represents the category to which the event belongs.

Syntax

[Visual Basic] Group As EventGroup
[C++\COM] HRESULT get_Group([out,retval] IEventGroup **ppVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only. See IEventGroups::ID and IEventGroups::Description for documentation on the type of this property.

IEventNotification::History

A multi-line string that details the history of the event or alarm.

Syntax

[Visual Basic] History As String
[C++\COM] HRESULT get_History([out,retval] BSTR *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only.

IEventNotification::Message

The message that would be displayed in the event window.

Syntax

[Visual Basic] Message As String
[C++\COM] HRESULT get_Message([out,retval] BSTR *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only.

IEventNotification::Priority

The priority of the event or alarm.

Syntax

[Visual Basic] Priority As Long
[C++\COM] HRESULT get_Priority([out,retval] Long *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only, and ranges from 1 to 9. 1 is the lowest priority, and 9 is the highest.

IEventNotification::Processed

Whether an alarm has been processed

Syntax

[Visual Basic] Processed As Long
[C++\COM] HRESULT get_Processed([out,retval] BOOL *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only. If the value of this property is 1 then the Event Notification represents an alarm that has been processed, otherwise the value of this property will be 0.

IEventNotification::Properties

Whether an alarm has been processed.

Syntax

[Visual Basic] Properties As CustomPropertyBag
[C++\COM] HRESULT get_Properties([out,retval] ICustomPropertyBag **ppVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only. The properties property should contain a number of the following properties.

Property	Type	Description
"Cardholder"	VT_BSTR	The name of the cardholder (card events only)
"EntryZone"	VT_BSTR	The name of the Entry zone (card events only)
"CardNumber"	VT_I4	The Card Number of the card (card events only)
"FacilityCode"	VT_I4	The Facility Code of the card (card events only)
"IssueLevel"	VT_I4	The Issue level of the card (card events only)
"Operator"	VT_BSTR	The name of the operator (operator events only)

IEventNotification::Refresh

Whether the notification is in response to a call to ISubscription::Refresh

Syntax

[Visual Basic] Refresh As Long
[C++\COM] HRESULT get_Refresh([out,retval] BOOL *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only. If the value of this property is 1 then the Event Notification is in response, otherwise the value of this property will be 0.

IEventNotification::Source

The source of the event.

Syntax

[Visual Basic] Source As String
[C++\COM] HRESULT get_Source([out,retval] BSTR *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only.

IEventNotification::Time

When the event Occurred.

Syntax

[Visual Basic] Time As Date
[C++\COM] HRESULT get_Time([out,retval] DATE *pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This Property is read only.

ILicence::Server

Name of the server for the licence to connect to, to determine whether a feature is licensable.

Syntax

[Visual Basic] Server As String
[C++\COM] HRESULT get_Server([out,retval] BSTR *pVal);
HRESULT put_Server([in] BSTR newVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Examples

See the ILicence::IsFeatureAvailable Method for an example use of this Property.

ISubscription::AttemptsBeforeNotify

The number of attempts before the Subscription will notify the subscribing object when An Event Server fails to connect.

Syntax

[Visual Basic] AttemptsBeforeNotify As Long
[C++\COM] HRESULT get_AttemptsBeforeNotify([out,retval] Long*pVal);
HRESULT put_AttemptsBeforeNotify([in] Long newVal);

Returns

S_OK	Successful Completion
E_FAIL	The Subscription object is currently active, and a change was attempted.
E_INVALIDARG	The New value is not between 1 and 999
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

When assigning this property the new value must be between 1 and 999. The put function will also fail when the Subscription object is currently active.

ISubscription::CheckEnabled

Whether the subscription should check whether the connection is still active.

Syntax

[Visual Basic] CheckEnabled As Long
[C++\COM] HRESULT get_CheckEnabled([out,retval] BOOL*pVal);
HRESULT put_CheckEnabled([in] BOOL newVal);

Returns

S_OK	Successful Completion
E_FAIL	The Subscription object is currently active
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

The put function will fail if the Subscription object is currently active. If the subscription is to check the connections, then the CheckEnabled property should be 1, otherwise the CheckEnabled property should be 0.

ISubscription::CheckInterval

The number of seconds that the subscription will wait between checks to determine whether the connection to any given event server is still active.

Syntax

[Visual Basic] CheckInterval As Long
[C++\COM] HRESULT get_CheckInterval([out,retval] Long*pVal);
HRESULT put_CheckInterval([in] Long newVal);

Returns

S_OK	Successful Completion
E_FAIL	The Subscription object is currently active, and a change was attempted.
E_INVALIDARG	The New value is not between 1 and 999
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

When assigning this property the new value must be between 1 and 999. The put function will also fail when the Subscription object is currently active.

ISubscription::ConnectAttempts

The maximum number of attempts that the Subscription should attempt every ConnectInterval Seconds.

Syntax

[Visual Basic] ConnectAttempts As Long
[C++\COM] HRESULT get_ConnectAttempts([out,retval] Long*pVal);
HRESULT put_ConnectAttempts([in] Long newVal);

Returns

S_OK	Successful Completion
E_FAIL	The Subscription object is currently active, and a change was attempted.
E_INVALIDARG	The New value is not between 1 and 99
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

When assigning this property the new value must be between 1 and 99. The put function will also fail when the Subscription object is currently active.

ISubscription::ConnectInterval

The size in seconds of the interval between which the subscription should attempt to connect up to ConnectAttempts times.

Syntax

[Visual Basic] ConnectInterval As Long
[C++\COM] HRESULT get_ConnectInterval([out,retval] Long*pVal);
HRESULT put_ConnectInterval([in] Long newVal);

Returns

S_OK	Successful Completion
E_FAIL	The Subscription object is currently active, and a change was attempted.
E_INVALIDARG	The New value is not between 1 and 999
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

When assigning this property the new value must be between 1 and 999. The put function will also fail when the Subscription object is currently active.

ISubscription::Enabled

Whether the Subscription is currently Active.

Syntax

[Visual Basic] Enabled As Long
[C++\COM] HRESULT get_Enabled([out,retval] Long*pVal);

Returns

S_OK	Successful Completion
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

This property is read only, if the subscription is active, the value will be 1 otherwise the value will be 0.

ISubscription::MaximumPriority

The Maximum Priority of an event that should get through the filter.

Syntax

[Visual Basic] MaximumPriority As Long
[C++\COM] HRESULT get_MaximumPriority([out,retval] Long*pVal);
HRESULT put_MaximumPriority([in] Long newVal);

Returns

S_OK	Successful Completion
E_FAIL	The Subscription object is currently active, and a change was attempted.
E_INVALIDARG	The New value is not between 1 and 9
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

When assigning this property the new value must be between 1 and 9. The put function will also fail when the Subscription object is currently active.

ISubscription::MinimumPriority

The Minimum Priority of an event that should get through the filter.

Syntax

[Visual Basic] MinimumPriority As Long
[C++\COM] HRESULT get_MinimumPriority([out,retval] Long*pVal);
HRESULT put_MinimumPriority([in] Long newVal);

Returns

S_OK	Successful Completion
E_FAIL	The Subscription object is currently active, and a change was attempted.
E_INVALIDARG	The New value is not between 1 and 9
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

When assigning this property the new value must be between 1 and 9. The put function will also fail when the Subscription object is currently active.

ISubscription::NotifyOnFailure

Whether the subscription should check whether the connection is still active.

Syntax

[Visual Basic] NotifyOnFailure As Long
[C++\COM] HRESULT get_NotifyOnFailure([out,retval] BOOL*pVal);
HRESULT put_NotifyOnFailure([in] BOOL newVal);

Returns

S_OK	Successful Completion
E_FAIL	The Subscription object is currently active
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

The put function will fail if the Subscription object is currently active. If the subscription is to Notify when a the subscription fails to connect to an event server more than AttemptsBeforeNotify times in a row, then the NotifyOnFailure property should be 1, otherwise the CheckEnabled property should be 0.

ISubscription::Server

The server that the Subscription should attempt to connect to.

Syntax

[Visual Basic] Server As String
[C++\COM] HRESULT get_Server([out,retval] BSTR*pVal);
HRESULT put_Server([in] BSTR newVal);

Returns

S_OK	Successful Completion
E_FAIL	The Subscription object is currently active
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

The put function will fail if the Subscription object is currently active. If this value does not contain a valid name of an Event server, then the GetAvailableGroups, Activate and LoadConfiguration member functions will not function correctly.

ISubscription::UTCTime

Whether the event occurrence times should be reported in UTC Time.

Syntax

[Visual Basic] UTCTime As Long
[C++\COM] HRESULT get_UTCTime([out,retval] BOOL*pVal);
HRESULT put_Server([in] BOOL newVal);

Returns

S_OK	Successful Completion
E_FAIL	The Subscription object is currently active
E_PENDING	Another member function of the Subscription object or one of it's children is currently executing.

Remarks

The put function will fail if the Subscription object is currently active.