Legacy Host Logging vs New Converter Endpoint
==========

1. Existing Service: Private Function Sending(...)

- Uses hostAdaptor.SendAndReceive(sSendMsg, sRecvMsg).
- Captures timestamps:
  sSendTime = Now.ToString("MM/dd/yyyy HH:mm:ss.ffff")
  sRecvTime = Now.ToString("MM/dd/yyyy HH:mm:ss.ffff")
- Builds sTraceLog based on HostMsgLogTrace setting (sendonly / receiveonly / sendandreceive):
  * sendonly   -> logs ONLY request (sSendMsg)
  * receiveonly   -> logs ONLY response (sRecvMsg)
  * sendandreceive-> logs BOTH request and response
- Finally, if sTraceLog.Length > 0 it calls:
  RaiseEvent LogError(msDBID, sTraceLog, "Trace to/from Host")
  This event handler ultimately writes the trace into the Encore SQL log table.

Conclusion: any call that goes through Sending(...) will persist the request/response pair to the Encore SQL logging ta

2. New Service: Private Sub SendToConverterEndpoint(ByRef xml As String, ByRef response As String)

- Reads MessageConverterURL from config.
- Reads HostMsgLogTrace into a local StringBuilder called "trace".
- Performs HTTP POST with HttpClient to the new converter endpoint.
- On success:
  - Sets "response" to resp.Content.ReadAsStringAsync().Result
  - Does NOT write any trace logs.
- On failure:
  - If trace is not Constants.Trace.None, it calls:
    RaiseEvent LogError("encore", ex.Message, ROUTINE)
  So only errors are logged, not the normal request/response XML.

Conclusion: calls going through SendToConverterEndpoint(...) are currently NOT logging the XML request/response p

3. How to Add Request/Response Logging to SendToConverterEndpoint

Goal: Make the new HTTP-based path behave like the legacy Sending(...) function, driven by the same HostMsgLogT

High-level steps:

1) Capture timestamps around the HTTP call:
  sendTime = Now.ToString("MM/dd/yyyy HH:mm:ss.ffff")
  recvTime = Now.ToString("MM/dd/yyyy HH:mm:ss.ffff")

2) After a successful response, build a traceLog string similar to Sending(...),
  based on HostMsgLogTrace (sendonly, receiveonly, sendandreceive). For example:

  - sendonly:
    "Send to Converter; len={xml.Length} {sendTime}\r\n{xml}"

  - receiveonly:
    "Receive from Converter; len={response.Length} {recvTime}\r\n{response}"

- sendandreceive:
  "Send to Converter; len={xml.Length} {sendTime}\r\n{xml}\r\n" +
  "Receive from Converter; len={response.Length} {recvTime}\r\n{response}"

3) If traceLog is not empty and HostMsgLogTrace is not "none", raise the same logging event:

RaiseEvent LogError("encore", traceLog, "Trace to/from Converter")

(You can also use msDBID instead of "encore" if you want it to behave exactly like the original.)

4. Sketch of Updated SendToConverterEndpoint (VB.NET)

```
Private Sub SendToConverterEndpoint(ByRef xml As String, ByRef response As String)
    Const ROUTINE As String = "HostMsg.SendToSTEndpoint"
    Const LogTraceKey As String = "HostMsgLogTrace"

    Dim trace As New StringBuilder()
    Dim endpointUrl As String = ConfigurationManager.AppSettings.Get("MessageConverterURL")

    Dim sendTime As String = Now.ToString("MM/dd/yyyy HH:mm:ss.ffff")
    Dim recvTime As String = Nothing

    Try
        trace.Append(ConfigurationManager.AppSettings.Get(LogTraceKey))
        If trace.Length = 0 Then
            trace.Append(Constants.Trace.None)
        End If

        ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12

        Using client As New HttpClient()
            client.Timeout = TimeSpan.FromSeconds(30)

            Dim content As New StringContent(xml, Encoding.UTF8, "application/xml")
            Dim resp As HttpResponseMessage = client.PostAsync(endpointUrl, content).Result
            response = Task.Run(Function() resp.Content.ReadAsStringAsync()).Result
            recvTime = Now.ToString("MM/dd/yyyy HH:mm:ss.ffff")

            If Not resp.IsSuccessStatusCode Then
                Throw New HttpRequestException(String.Format("{0} Error {1}: {2}", ROUTINE, CInt(resp.StatusCode), resp
            End If
        End Using

        ' --- request/response logging similar to Sending ---
        Dim traceSetting As String = trace.ToString().ToLowerInvariant()
        Dim traceLog As String = String.Empty

        If traceSetting = "sendonly" Then
            traceLog = String.Format("Send to Converter; len={1}{0}{2}{3}", sendTime, xml.Length & " ", vbCrLf, xml)
        ElseIf traceSetting = "receiveonly" Then
            traceLog = String.Format("Receive from Converter; len={1}{0}{2}{3}", recvTime, response.Length & " ", vbCrLf,
        ElseIf traceSetting = "sendandreceive" Then
            traceLog = String.Format("Send to Converter; len={1}{0}{2}{3}", sendTime, xml.Length & " ", vbCrLf, xml) & _
                       vbCrLf & _
```

```
                String.Format("Receive from Converter; len={1}{0}{2}{3}", recvTime, response.Length & " ", vbCrLf, resp
        End If

        If traceLog.Length > 0 AndAlso Not traceSetting.Equals(Constants.Trace.None, StringComparison.OrdinalIgnore
            RaiseEvent LogError("encore", traceLog, "Trace to/from Converter")
        End If

    Catch ex As Exception
        If trace.Length > 0 AndAlso trace.ToString().Equals(Constants.Trace.None) = False Then
            RaiseEvent LogError("encore", ex.Message, ROUTINE)
        End If
    End Try
End Sub
```

5. Quick Summary

- Old path Sending(...) → logs request/response to Encore SQL via LogError event when HostMsgLogTrace is configu
- New path SendToConverterEndpoint(...) → currently logs only errors, not normal traffic.
- To keep observability and debugging parity, add a traceLog section after the successful HTTP call and raise the sam