



Cognitier SimpleIPC

Client Code Sample – VB.NET

Version 1.0.0.1

April 21, 2009

Copyright © 2009 Cognitier, Inc. All rights reserved.

Cognitier and the Cognitier logo are trademarks of Cognitier, Inc. All other company and product names referred to may be trademarks of their respective owners. This documentation is copyrighted material and is intended exclusively for use by licensed users of Cognitier software. The information in this document is subject to change without notice.

Client Code Sample – VB.NET

The following sample client code is provided to demonstrate the use of the most common SimpleIPC client API calls in a .NET programming environment. The sample code below is written in VB.NET. The code would be similar for C#. There are some differences in the APIs used if you write your code in Jscript.NET – this is covered in the product documentation. In this example, your application is the client, and it is written in VB.NET. You want to invoke a custom server program you recently wrote to be hosted by SimpleIPC.

In the following sample code, the SimpleIPC client object is instantiated. You can uncomment some of the code below to get a reference to the SimpleIPC COM object and use the COM API, rather than the .NET API, from your .NET client if you prefer. After the client object is instantiated, arguments are passed via the object to a remote object running in an IPC server, and the remote object is instructed to run a custom piece of code that you, the software developer, would have previously written. You may have any of several reasons for not wanting the code in question to run in your client's process space. When the call returns, there is an array of output values that was prepared by the remote procedure call. The sample code invokes the remote routine several times in a loop for demonstration purposes. It also illustrates checking the return value from each call to see if there was a failure in the remote process. If so, the code obtains a new session (presumably with a new remote process) and resumes processing.

```
Module Module1

    Sub Main()
        'Dim bNeedToReleaseComp As Boolean
        'Dim oComp As Interop.CTOutProcCOMInst1Lib.CTSessionComp1
        Try
            'Declare variables
            Dim oSessionEntry As New CTRClient.SessionEntry
            Dim sSessionID As String = String.Empty
            Dim iRejectCode As Integer
            Dim sRejCode As String = String.Empty
            Dim sInstanceName As String = "Inst1"
            Dim sAppID As String = "App1"
            Dim sRoutine as String = "YourTestRoutineHere"
            Dim oInputAL As New ArrayList
            Dim oOutputAL As New ArrayList
            Dim oErrorAL As New ArrayList
            Dim oWarningAL As New ArrayList
        End Try
    End Sub
End Module
```

```

Dim iElemCount As Integer
Dim bRet As Boolean
Dim iNumCallsToMake As Integer = 3
Dim iCallCount As Integer
Dim bIncrementCallCount As Boolean

'Initialize variables
oInputAL.Add("Sample Input 1")
oInputAL.Add("Sample Input 2")
oInputAL.Add("Sample Input 3")

'Make call to obtain a session
bRet = oSessionEntry.ObtainSession(sSessionID, sInstanceName,
sAppID, iRejectCode)

.....
'''
    'The call to obtain session may require a server to be started.
    'If your application runs under a user account which does not have
execute permission
    'in the \Cognitier\SimpleIPC\bin directory, then set the DCOM RunAs
user to one
    'having this permission and use the COM object to obtain the
session id.
    'After obtaining the session id via the COM object, the
SessionEntry .NET object
    'can be used to make subsequent calls.

    'oComp = New Interop.CTOutProcCOMInst1Lib.CTSessionComp1
    'bNeedToReleaseComp = True
    'bRet = oComp.ObtainSession(sSessionID, sAppID, iRejectCode)

.....
'''

If bRet = False Then
    Try
        sRejCode = CType(iRejectCode,
CTCommon.Constants.ObtainSessionRejectReason).ToString
    Catch exRejCode As Exception
        'Do Nothing
    End Try

```

```

        Console.WriteLine("ObtainSession return = " & bRet & "; rej code
= " & iRejectCode & " (" & sRejCode & ")")
    Else
        Console.WriteLine("Session obtained; session id = " &
sSessionID)
        'For demonstration purposes, make several routine calls in a
loop
        Do While iCallCount < iNumCallsToMake
            bIncrementCallCount = True
            'Make call to BasicRoutineCall1
            bRet = oSessionEntry.BasicRoutineCall1(sSessionID,
sInstanceName, sAppID, sRoutine, oInputAL, oOutputAL, oErrorAL, oWarningAL)
            If bRet = False Then
                If oErrorAL.Count > 0 Then
                    Try
                        If CInt(oErrorAL(0)) =
CTCommon.Constants.RoutineExecutionErrors.SessionInvalidOrTimedOut Or
CInt(oErrorAL(0)) =
CTCommon.Constants.RoutineExecutionErrors.ServerCommunicationErr Or
CInt(oErrorAL(0)) =
CTCommon.Constants.RoutineExecutionErrors.ErrResumingSession Then
                            'The call to BasicRoutineCall1 failed, but
the error code indicates there might be a problem with the session, so try to
get a new session
                                sSessionID = ""
                                iRejectCode = 0
                                bRet =
oSessionEntry.ObtainSession(sSessionID, sInstanceName, sAppID, iRejectCode)
                                .....
                                'Again, it may be necessary to use the COM
object to obtain the session
                                    bRet = oComp.ObtainSession(sSessionID,
sAppID, iRejectCode)
                                    .....
                                If bRet = False Then
                                    Exit Do
                                Else
                                    bIncrementCallCount = False
                                End If
                            Else

```

```

        Exit Do
    End If
    Catch exCheckErr As Exception
        Exit Do
    End Try
Else
    Exit Do
End If
Else
    iElemCount = 0
    Do While iElemCount < oOutputAL.Count
        Console.WriteLine("Call " & iCallCount & "; Output
Array Elem " & iElemCount & " = " & oOutputAL(iElemCount))
        iElemCount = iElemCount + 1
    Loop

    iElemCount = 0
    Do While iElemCount < oErrorAL.Count
        Console.WriteLine("Call " & iCallCount & "; Error
Array Elem " & iElemCount & " = " & oErrorAL(iElemCount))
        iElemCount = iElemCount + 1
    Loop

    iElemCount = 0
    Do While iElemCount < oWarningAL.Count
        Console.WriteLine("Call " & iCallCount & "; Warning
Array Elem " & iElemCount & " = " & oWarningAL(iElemCount))
        iElemCount = iElemCount + 1
    Loop
End If
'Increment the call count unless the previous call failed
and we had to get a new session
If bIncrementCallCount = True Then
    iCallCount = iCallCount + 1
End If
Loop
If iCallCount < (iNumCallsToMake - 1) Then
    Console.WriteLine("Unable to complete all routine calls")
End If

'Release the session

```

```

        oSessionEntry.ReleaseSession(sSessionID)
        oSessionEntry.Dispose()
    End If

    'Release the COM object - if you used it
    'While
System.Runtime.InteropServices.Marshal.ReleaseComObject(oComp) > 0
    'End While
    'bNeedToReleaseComp = False

    Catch ex As Exception
        Console.WriteLine("Exception encountered: " & ex.Message)
        'If bNeedToReleaseComp = True Then
            '    While
System.Runtime.InteropServices.Marshal.ReleaseComObject(oComp) > 0
            '    End While
        'End If
    End Try

    Console.WriteLine("Execution complete")
End Sub

End Module

```