Part 12 Interface – Using the CAA Occurrences Web Service

Intended Audience: Software Developers

Version 1.5

Contents

Document Revision History

Date	Document Version	Updated by	Description of changes
24.03.2010	0.1	Michael Heitland	Initial creation.
31.03.2010	1.0	Michael Heitland	Description of test programs added
06.04.2010	1.1	Michael Heitland	Screenshot added, proxy configuration
06.04.2010	1.2	Michael Heitland	Links to schemas updated
07.04.2010	1.3	Michael Heitland	Links to schemas updated
22.04.2010	1.4	Michael Heitland	Process diagram and text updated: Service returns either acceptance message or a FaultException with error details
4.5.2010	1.5	Michael Heitland	FaultException explained with envelope

Related Documents

Document	Description
http://www.caa.govt.nz/Accidents_and_Incidents/Electronic_Reporting/Attachment_1_0.pdf	Attachment Schema (PDF Document)
http://www.caa.govt.nz/Accidents_and_Incidents/Electronic_Reporting/Attachment_1_0.xsd	Attachment Schema (XML Schema)
http://www.caa.govt.nz/Accidents_and_Incidents/Electronic_Reporting/Attachment_1_0.xml	Sample Attachment Data (XML Data)
http://www.caa.govt.nz/Accidents_and_Incidents/Electronic_Reporting/ClosedActions_1_0.pdf	ClosedActions Schema (PDF Document)
http://www.caa.govt.nz/Accidents_and_Incidents/Electronic_Reporting/ClosedActions_1_0.xsd	ClosedActions Schema (XML Schema)
http://www.caa.govt.nz/Accidents_and_Incidents/Electronic_Reporting/ClosedActions_1_0.xml	Sample ClosedActions Data (XML Data)
http://www.caa.govt.nz/Accidents_and_Incidents/Electronic_Reporting/Occurrence_1_0.pdf	Occurrence Schema (PDF Document)
http://www.caa.govt.nz/Accidents_and_Incidents/Electronic_Reporting/Occurrence_1_0.xsd	Occurrence Schema (XML Schema)
http://www.caa.govt.nz/Accidents_and_Incidents/Electronic_Reporting/Occurrence_1_0.xml	Sample Occurrence Data (XML Data)
http://www.caa.govt.nz/Accidents_and_Incidents/Electronic_Reporting/Descriptors_1_0.xlsx	CAA Descriptors (Excel Table)
http://www.caa.govt.nz/Accidents_and_Incidents/Electronic_Reporting/CAA.Part12Interface.CaptureService.Client.Setup.zip	Part12 Capture Service Client (Demo Application)

Overview

The purpose of this solution is to provide clients with the ability to submit Rule Part 12 occurrence information (including Investigations, Findings, Causes and Actions) through an XML data interface to CAA. It is replacing a previous solution that received the same data as Microsoft Access databases or attached Word documents by email.

Data Flow for Capturing Part 12 Occurrences

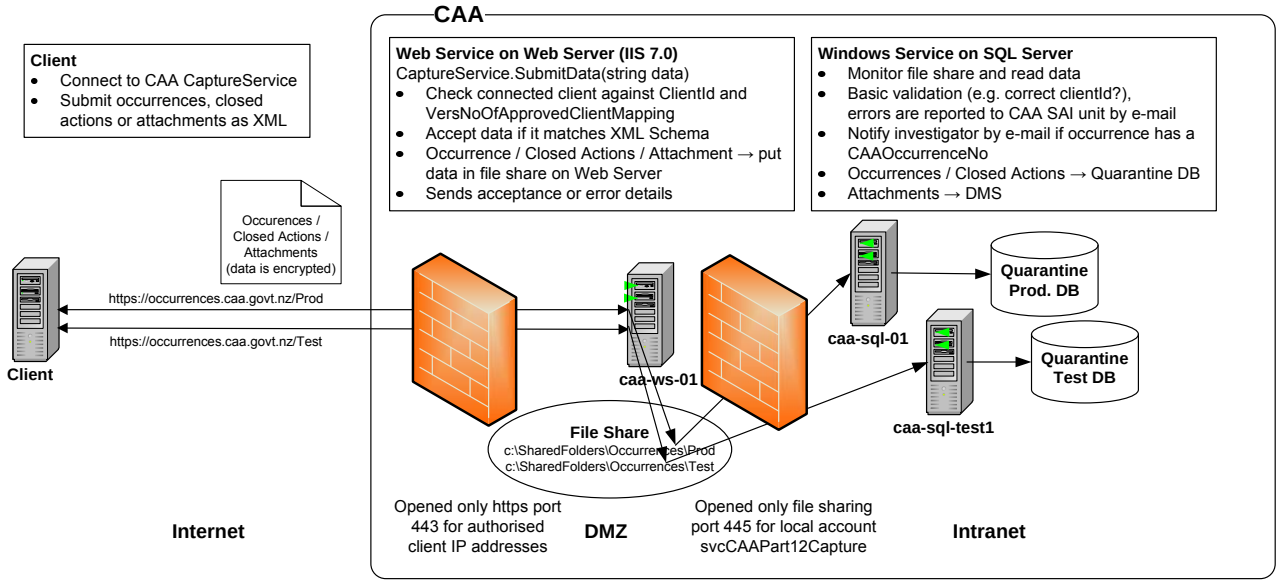


Figure 1 –Data Flow (see Appendix A – Data Flow for Capturing Part 12 Occurrences for a magnified view)

The CAA Part 12 Occurrences Service is a .NET WCF web service with only one function: receiving XML data as a string and storing it into its local file share.

```
public void SubmitData(string data)
```

It accepts data (occurrences, findings, causes, actions, closed actions and attachments) as an XML string. After data has been received, it is validated and stored as a file on a local file share for further processing and an acceptance is immediately sent back to the caller (empty message envelope). In case of errors, a FaultException<CaptureServiceFault> message is returned (message envelope containing the error details inside).

These are the processing steps of the Occurrences Service:

- Check connected client against ClientId and VersNoOfApprovedClientMapping
- Accept data if it matches XML Schema, otherwise return an error message to client
- Occurrence / Closed Actions / Attachment: put data into file share on Web Server

The web service is hosted on IIS 7.0 on a web server sitting in a DMZ.

We are using a TLS WsHttp connection with an X.509 certificate on our web server site occurrences.caa.govt.nz. Only authorized clients that have undergone a sign-in process and passed their IP address to CAA can connect with the service using the following link:

https://occurrences.caa.govt.nz/Prod (for real data going into CAA production system) or
https://occurrences.caa.govt.nz/Test (for test data going into CAA test system).

Implementing the Consumer Side of the CAA Occurrences Web Service

The CAA Occurrences web service was implemented using .NET Windows Communication Framework (WCF). So it is based on open standards and can easily be consumed by any IT technology that conforms to W3C standards for web services (.NET, Java etc.).

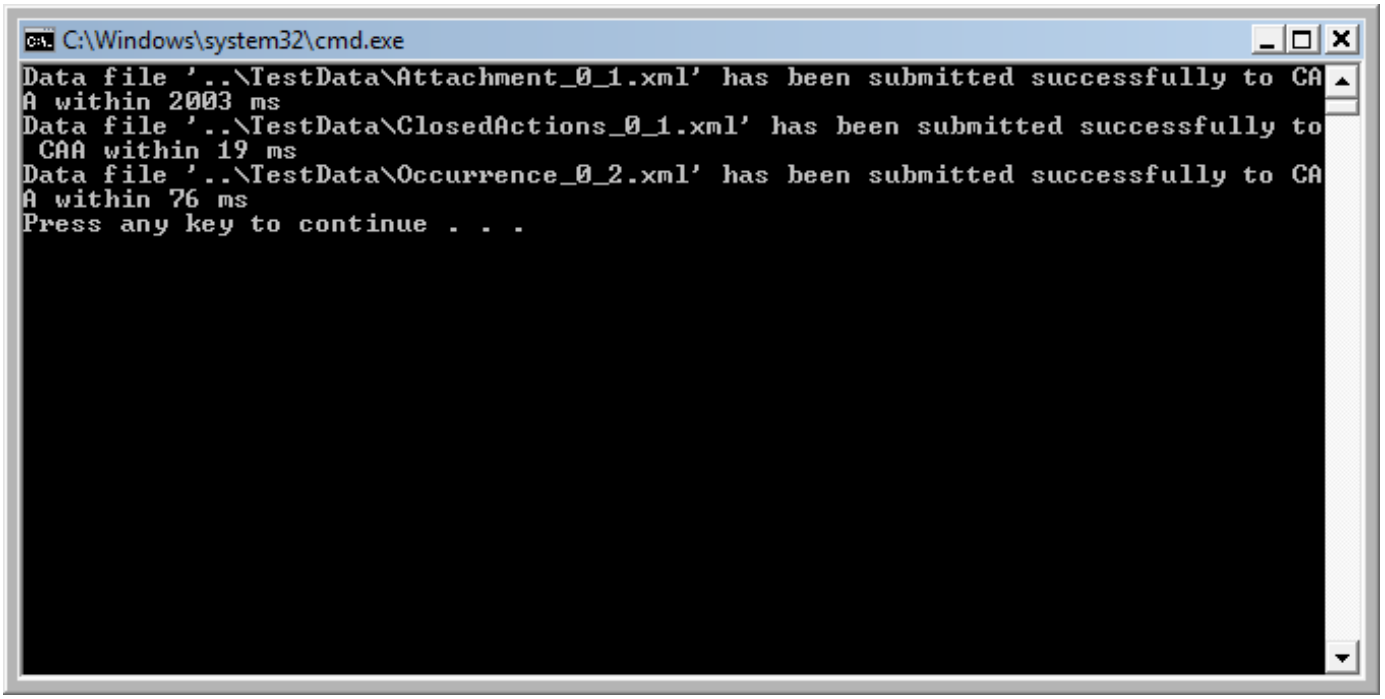
Client Demo Program

You can find a basic .NET implementation in Appendix C – Example of a CAA Occurrences Service Client.The binaries of that CAA Occurrences client can be downloaded from https://www.caa.govt.nz/Accidents_and_Incidents/electronic_reporting.htm.
After extracting the files and running setup.exe you can submit some sample occurrences, closed actions and attachments to CAA by running

C:\Program Files\Civil Aviation Authority\CAA.Part12Interface.CaptureService.Client\Binaries\TestSubmitData.bat

The default setting for this little program is not use a proxy, if you are running that demo program behind a proxy, you will get an error message “There was no endpoint listening at ...”. In this case you have to change the config settings to use your proxy (see Appendix C).

After submitting data successfully the demo program shows the following screen:



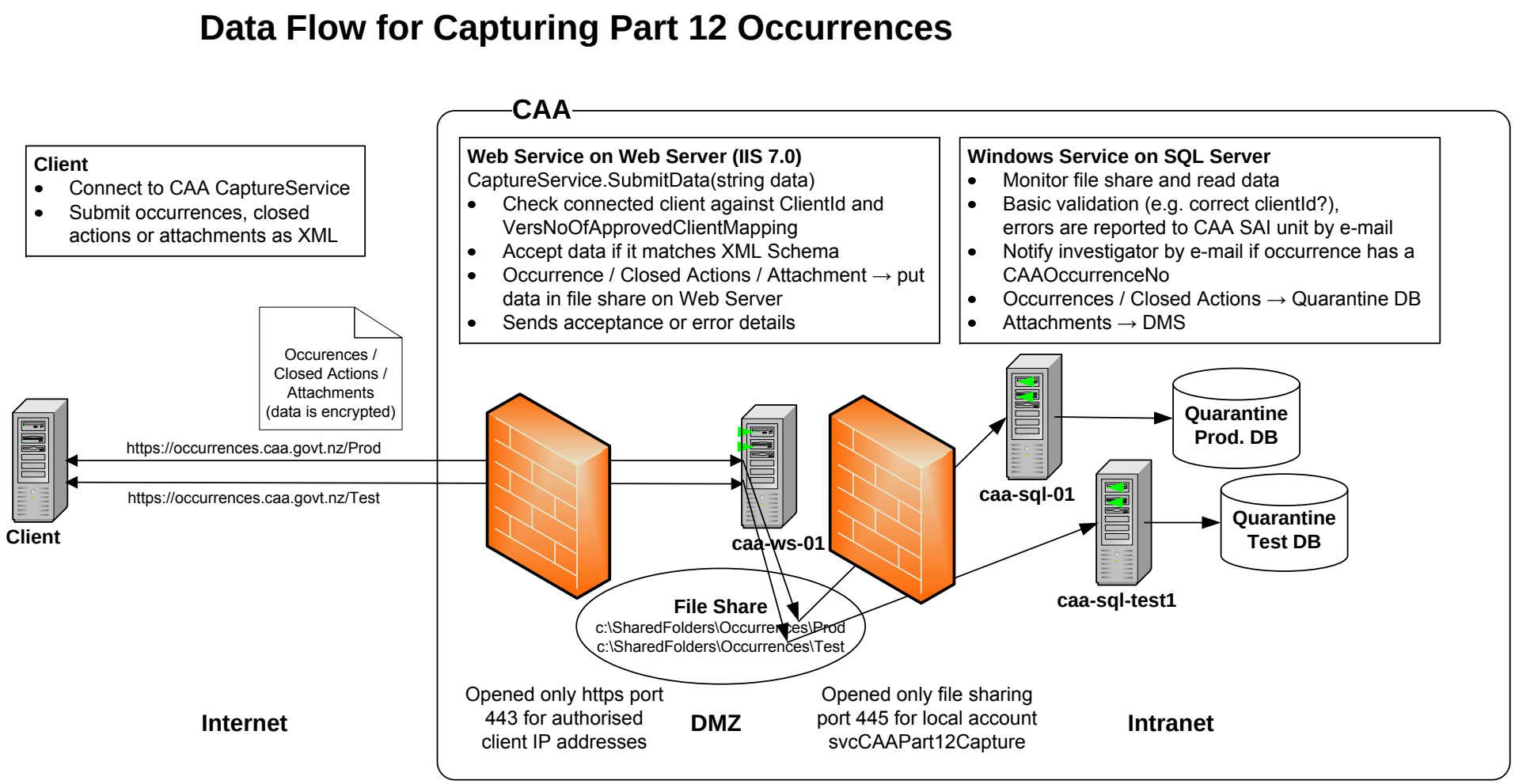
In the same folder you will find another program to convert any file into an attachment.xml file that can be send to CAA by running TestAttachmentConverter.bat. Just make sure that you have writing permissions to the output folder.

Implementation Details

A client would have to implement the following steps:

- Create or get the data (occurrences, closed actions or attachment) in XML form
- Make sure that the data is matching its CAA XML schema
  (e.g. attachment data has to match Attachment_1_0.xsd)
- Open an https connection to the CAA Occurrences web service:
  https://occurrences.caa.govt.nz/Prod (for real data going into CAA production system)
  or https://occurrences.caa.govt.nz/Test (for test data going into CAA test system).
- Send data (occurrences, closed actions and attachments) as an XML string to CAA:
  `service.SubmitData(data)`
  For details see Appendix B – Interface ICaptureService
- Files that have been previously sent to CAA as mail attachments (e.g. `report.doc`) are now also transferred as XML data via `service.SubmitData(data)`.
  The file content has to be put into the `base64Binary` encoded field *FieldData* , for details see the attachment schema. The file size for an attachment is restricted and must not exceed 10 MB.
- Close https connection

Appendix A – Data Flow for Capturing Part 12 Occurrences



Appendix B – Interface ICaptureService

```
using System;
using System.Runtime.Serialization;
using System.ServiceModel;

namespace CAA.Part12Interface.CaptureService
{
    [ServiceContract(
        Namespace = "http://www.caa.govt.nz/CAA.Part12Interface.CaptureService")]
    public interface ICaptureService
    {
        /// <summary>
        /// send occurrence report data to CAA capture service
        /// (occurrences, attachments, closed actions)
        /// IMPORTANT: XML data must conform to schema!
        /// </summary>
        /// <param name="data">occurrence report data in XML format</param>
        [OperationContract]
        [TransactionFlow(TransactionFlowOption.Allowed)]
        [FaultContract(typeof(CaptureServiceFault))]
        void SubmitData(string data);
    }

    [DataContract(
        Namespace = "http://www.caa.govt.nz/CAA.Part12Interface.CaptureService")]
    public class CaptureServiceFault
    {
        [DataMember]
        public string UserMessage { get; private set; }

        [DataMember]
        public string DeveloperMessage { get; private set; }

        public CaptureServiceFault(string userMessage, Exception exception)
        {
            UserMessage = userMessage;
            DeveloperMessage = userMessage + Environment.NewLine + exception;
        }
    }
}
```

Appendix C – Example of a CAA Occurrences Service Client

CAA Occurrences Service Client sending XML data to CAA

```
//Add this web service reference to project:
//https://occurrences.caa.govt.nz/Test/CAA.Part12Interface.CaptureService.CaptureService.svc
using System;
using System.Diagnostics;
using System.IO;
using System.Net;
using System.Net.Security;
using System.Security.Cryptography.X509Certificates;
using System.ServiceModel;
using CAA.Part12Interface.CaptureService.Client.CaptureServiceReference;

namespace CAA.Part12Interface.CaptureService.Client
{
    class CaptureServiceProfiler
    {
        // helps us to accept self-assigned certificates
        private static bool IgnoreCertificateErrorHandler(
            object sender, X509Certificate certificate, X509Chain chain,
            SslPolicyErrors sslPolicyErrors)
        {
            return true;
        }

        static void Main(string[] args)
        {
            try
            {
                var folderPathSchema = Properties.Settings.Default.FolderPathSchema;
                var altFolderPathSchema =
                      Properties.Settings.Default.AltFolderPathSchema;
                if (!Directory.Exists(folderPathSchema))
                {
                    if (!Directory.Exists(altFolderPathSchema))
                        throw new Exception(string.Format(
                            "Cannot find schema folder '{0}' or ''{1}",
                            folderPathSchema,
                            altFolderPathSchema));
                    folderPathSchema = altFolderPathSchema;
                }
                if (args.Length == 0)
                    throw new Exception(
                        "Usage: CAA.Part12Interface.CaptureService.Client.exe " +
                        "<filePath> ...\r\n" +
                        "Example: CAA.Part12Interface.CaptureService.Client.exe " +
                        "Occurrence_1_0");

                ServicePointManager.ServerCertificateValidationCallback =
                  new RemoteCertificateValidationCallback(IgnoreCertificateErrorHandler);

                using (var clientCaptureService = new CaptureServiceClient())
                {
                    foreach (var filePathData in args)
                    {
                        // load data from file into memory and check data against schema
                        var captureData = new CaptureData.CaptureData(
                           folderPathSchema, filePathData);
```

```
                        var stopwatch = new Stopwatch();
                        stopwatch.Reset();
                        stopwatch.Start();
                        clientCaptureService.SubmitData(captureData.Data);
                        stopwatch.Stop();
                        Console.WriteLine(string.Format(
                            "Data file '{0}' has been submitted successfully to CAA " +
                            "within {1} ms",
                            filePathData, stopwatch.ElapsedMilliseconds));
                    }
                }
                //System.Threading.Thread.Sleep(3000);
            }
            catch (FaultException<CaptureServiceFault> ex)
            {
                Console.WriteLine("string.Format(*** Error (UserMessage):\r\n{0}\r\n",
                    ex.Detail.UserMessage);
                Console.WriteLine("string.Format(*** Error (DeveloperMessage:\r\n{0}",
                    ex.Detail.DeveloperMessage);
            }
            catch (Exception ex)
            {
                Console.WriteLine("*** Error: " + ex);
            }
        }
    }
}
```

Helper Class CaptureData in CaptureData.cs

```
#region Using

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Xml.Linq;

using CAA.Common.XML;

#endregion Using

namespace CAA.Part12Interface.CaptureService.CaptureData
{
    public class CaptureData
    {
        #region Constants

        private static readonly List<string> ValidFileExtensions =
            new List<string> { ".att", ".clo", ".occ" };

        #endregion Constants

        #region Properties

        public string Data { private set; get; }

        public string XmlNamespace { private set; get; }

        public string SchemaVersion { private set; get; }

        public int ClientId { private set; get; }

        public string FileExt { private set; get; }

        public string VersNoOfApprovedClientMapping { private set; get; }

        #endregion Properties

        #region Constructor

        public CaptureData(string folderPathSchema, string filePathData)
        {
            if (!Directory.Exists(folderPathSchema))
                throw new Exception(string.Format(
                    "Cannot find schema folder '{0}'", folderPathSchema));
            if (!File.Exists(filePathData))
                throw new Exception(string.Format(
                    "Cannot find file '{0}'", filePathData));
            Data = File.ReadAllText(filePathData);
            if (string.IsNullOrEmpty(Data))
                throw new Exception(string.Format(
                    "File '{0}' is empty", filePathData));

            // extract XmlNamespace, fileExt, clientId and
            // versNoOfApprovedClientMapping from XML Data
            var el = XElement.Parse(Data); // throws an exception if Data is no valid XML
            if (el.Name == null || el.Name.Namespace == null ||
```

```
                string.IsNullOrEmpty(el.Name.NamespaceName))
                throw new Exception("xmlns missing");
            XmlNamespace = el.Name.NamespaceName;
            var pos = XmlNamespace.LastIndexOf('/');
            if (pos == -1 || pos == XmlNamespace.Length - 1)
                throw new Exception("xmlns has incorrect format");
            SchemaVersion = XmlNamespace.Substring(pos + 1);
            if (SchemaVersion.Length < 3)
                throw new Exception(string.Format(
                    "SchemaVersion '{0}' has incorrect format", SchemaVersion));
            FileExt = "." + SchemaVersion.Substring(0, 3).ToLower();
            if (FileExt.Length != 4 || !ValidFileExtensions.Contains(FileExt))
                throw new Exception(string.Format(
                    "xmlns has incorrect format, fileExt '{0}' has incorrect format " +
                    "or is not supported", FileExt));
            var clientIdStr = (
                from a in el.Elements()
                where a.Name.LocalName == "ClientId"
                select a.Value).FirstOrDefault();
            if (string.IsNullOrEmpty(clientIdStr))
                throw new Exception("ClientId missing");
            int clientId;
            if (!int.TryParse(clientIdStr, out clientId))
                throw new Exception("ClientId is non-numeric");
            ClientId = clientId;
            VersNoOfApprovedClientMapping =
                (from a in el.Elements()
                 where a.Name.LocalName == "VersNoOfApprovedClientMapping"
                 select a.Value).FirstOrDefault();
            if (FileExt == ".occ" && string.IsNullOrEmpty(VersNoOfApprovedClientMapping))
                throw new Exception(
                    "versNoOfApprovedClientMapping is mandatory for occurrence Data");

            // check Data against schema
            var filePathSchema = string.Format("{0}\\{1}.xsd",
                folderPathSchema,
                SchemaVersion);
            var schemaValidator = new SchemaValidator();
            schemaValidator.ValidateSchema(filePathSchema, filePathData);
        }

        #endregion Constructor
    }
}
```

Helper Class SchemaValidator in SchemaValidator.cs

```
using System;
using System.IO;
using System.Xml;
using System.Xml.Schema;

namespace CAA.Common.XML
{
    public class SchemaValidator
    {
        public void ValidateSchema(string filePathSchema, string filePathData)
        {
            try
            {
                if (string.IsNullOrEmpty(filePathSchema))
                    throw new ArgumentNullException("filePathSchema");
                if (string.IsNullOrEmpty(filePathData))
                    throw new ArgumentNullException("filePathData");
                if (!File.Exists(filePathSchema))
                    throw new Exception(string.Format(
                        "Cannot find XML schema file '{0}'", filePathSchema));
                if (!File.Exists(filePathData))
                    throw new Exception(string.Format(
                        "Cannot find XML data file '{0}'", filePathData));

                var settings = new XmlReaderSettings
                {
                    ValidationType = ValidationType.Schema,
                    ValidationFlags = XmlSchemaValidationFlags.ReportValidationWarnings
                };
                settings.Schemas.Add(null, filePathSchema);

                using (XmlReader r = XmlReader.Create(filePathData, settings))
                {
                    while (r.Read())
                    {
                        // xml file will be validated automaticallz while reading
                    }
                }
            }
            catch (Exception ex)
            {
                throw new Exception(string.Format(
                        "xml file '{0}' does not match schema '{1}'",
                    filePathData, filePathSchema), ex);
            }
        }
    }
}
```

Settings in CAA.Part12Interface.CaptureService.Client.exe.config

Important: if your client sits behind a proxy, change line

bindingConfiguration="CaptureServiceWithoutProxy"

to

bindingConfiguration="CaptureServiceWithProxy"

and change **proxyAddress="http://caa-isa:8080"** to use your own proxy.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <configSections>
        <sectionGroup name="applicationSettings"
         type="System.Configuration.ApplicationSettingsGroup, System, Version=2.0.0.0,
               Culture=neutral, PublicKeyToken=b77a5c561934e089" >
           <section name="CAA.Part12Interface.CaptureService.Client.Properties.Settings"
            type="System.Configuration.ClientSettingsSection, System, Version=2.0.0.0,
                  Culture=neutral, PublicKeyToken=b77a5c561934e089"
            requirePermission="false" />
        </sectionGroup>
    </configSections>

  <system.serviceModel>

    <bindings>
      <wsHttpBinding>
        <binding name="CaptureServiceWithoutProxy" closeTimeout="00:01:00"
          openTimeout="00:01:00"
          receiveTimeout="00:10:00" sendTimeout="00:10:00" bypassProxyOnLocal="false"
          transactionFlow="false" hostNameComparisonMode="StrongWildcard"
          maxBufferPoolSize="12345678" maxReceivedMessageSize="12345678"
          messageEncoding="Mtom" textEncoding="utf-8"
          useDefaultWebProxy="true" allowCookies="false">
          <readerQuotas maxDepth="12345678" maxStringContentLength="12345678"
            maxArrayLength="12345678" maxBytesPerRead="12345678"
            maxNameTableCharCount="12345678" />
          <reliableSession ordered="true" inactivityTimeout="00:10:00"
            enabled="false" />
          <security mode="Transport">
            <transport clientCredentialType="None" proxyCredentialType="None"
              realm="" />
            <message clientCredentialType="Windows" negotiateServiceCredential="true"
              establishSecurityContext="true" />
          </security>
        </binding>

        <binding name="CaptureServiceWithProxy" closeTimeout="00:01:00"
          openTimeout="00:01:00"
          receiveTimeout="00:10:00" sendTimeout="00:10:00" bypassProxyOnLocal="false"
          transactionFlow="false" hostNameComparisonMode="StrongWildcard"
          maxBufferPoolSize="12345678" maxReceivedMessageSize="12345678"
          messageEncoding="Mtom"
          proxyAddress="http://caa-isa:8080"
          textEncoding="utf-8"
          useDefaultWebProxy="false" allowCookies="false">
          <readerQuotas maxDepth="12345678" maxStringContentLength="12345678"
            maxArrayLength="12345678" maxBytesPerRead="12345678"
            maxNameTableCharCount="12345678" />
```

```
          <reliableSession ordered="true" inactivityTimeout="00:10:00"
            enabled="false" />
          <security mode="Transport">
            <transport clientCredentialType="None" proxyCredentialType="None"
              realm="" />
            <message clientCredentialType="Windows" negotiateServiceCredential="true"
              establishSecurityContext="true" />
          </security>
        </binding>
      </wsHttpBinding>
    </bindings>

    <client>
      <endpoint address="https://occurrences.caa.govt.nz/Test/CAA.Part12Interface.
                         CaptureService.CaptureService.svc"
                binding="wsHttpBinding"
                bindingConfiguration="CaptureServiceWithoutProxy"
                contract="CaptureServiceReference.ICaptureService"
                name="CaptureService">
        <identity>
          <dns value="localhost" />
        </identity>
      </endpoint>
    </client>

  </system.serviceModel>

    <applicationSettings>
        <CAA.Part12Interface.CaptureService.Client.Properties.Settings>
            <setting name="FolderPathSchema" serializeAs="String">
                <value>..\Schema</value>
            </setting>
            <setting name="AltFolderPathSchema" serializeAs="String">
                <value>..\..\..\Schema</value>
            </setting>
        </CAA.Part12Interface.CaptureService.Client.Properties.Settings>
    </applicationSettings>
</configuration>
```