

FACILITY ASSOCIATION

151 Yonge Street • 18th Floor • Toronto • Ontario • M5C 2W7
Tel: (416) 863-1750 • Fax: (416) 868-0894 • E-mail: mail@facilityassoc.com

TO: ALL MEMBERS OF THE FACILITY ASSOCIATION
ATTENTION: CHIEF EXECUTIVE OFFICER
ONTARIO POOL PROJECT MANAGER
BULLETIN NO.: F05-020
DATE: APRIL 14, 2005
SUBJECT: ONTARIO RISK SHARING POOL FACT IMPLEMENTATION
Update: ONTARIO FACT Application Project

The development of the **ONTARIO FACT WEB** application is underway and the project is progressing as planned. The Ontario FACT Test version will be available in early May 2005 and the production application is scheduled for release on June 1, 2005.

Although the applications are not yet available, the user id setup for both test and production application has begun. This is to avoid the rush on the setup of users just prior to the release of the applications. If you have requested an id you should receive the information within the next two weeks. However, if you have not yet requested access please do so as soon as possible to allow for the timely setup of your user id. The user id request form is attached below and is also available from Facility Association.

Current users who already have access to Alberta or New Brunswick FACT Web application please note that you will not be receiving a new production user id or password. However, to get access to Ontario FACT you are required to complete the user id request form which should include your current user id. Your access profile will be modified and on June 1st you will be able to access the Ontario FACT by selecting it from the application dropdown menu. If you have requested a new test id then you will be receiving an email with your test user id and password shortly.

New users who have submitted their id request form and asked for both test and production user ids will be receiving two emails with separate user information to both test and production environments.

To avoid confusion the test user id's start with the word "TEST". There will be no conversion of test id's into production ids. Test user id's can only be used for testing and will be disabled on July 1, 2005.

BULLETIN F05-020
ONTARIO RISK SHARING POOL FACT IMPLEMENTATION
UPDATE: ONTARIO FACT APPLICATION PROJECT

IMPORTANT: Companies currently using **cartridge** or mainframe **IIN mailbox** (formerly SNA) for their Ontario Risk Sharing Pool submissions please note that these submission methods will no longer be supported after May 31st, 2005. In the new Ontario FACT Web application you can automate the submission of batch data files via web service. For further information on the FACT Web service please review FACT Web Service Submission Guide attached below.

Starting June 1st new Risk Sharing Pool data submission will only be accepted through the FACT Web application. Outstanding errors can be corrected and submitted by either the PC Pool Package or by FACT during the transitional month of June 2005. All outstanding errors from the PC Pool Package should be corrected by June 30th, 2005. No PC Pool package corrections will be accepted after June 30th, 2005.

There will be further communication to the companies and users when the Ontario FACT Test application is released.

Georgette Chan
Vice President, Finance & Operations

Request Form (User Id & Branch Code)

Facility Association

151 Yonge Street, 18th Floor, Toronto, Ontario, Canada M5C 2W7
Tel.: (416) 863-1750 Fax.: (416) 868-0894

FA Correction & Transmission (FACT)

Instructions

Please **PRINT** clearly.

For all requests

- The Company Information section must be completed.
- Project Manager, must sign and fax back to (416) 842-0241

For user id request:

- The User Information section must be completed in full.
- The user and a witness must sign the security agreement.

For company branch requests:

- At least one branch code must be assigned to each IBC Reporting Company Number in the Branch Information section.

Company Information (please print)

Company Name: _____

FA Project Manager: _____

FA Project Manager Email Address: _____

User Information

User Name: _____ Title: _____

Branch Address _____

Phone: _____ Fax: _____

Email Address: _____

Please select one of the following options:

- New User ID
- More Access for Existing User ID # _____
- Cancel User ID # _____

PRODUCTION User Access required for: *

Transmission & Correction Administrator Co-ordinator

TEST User Access required for: *

Transmission & Correction Administrator Co-ordinator

FACT Web Services Access required for: **

TEST FACT Web Services PRODUCTION FACT Web Services

* A **Transmission & Correction User** will be able to enter, transmit and correct transactions.
An Administrator User will be able to view and revive previously transmitted batches.
A Co-ordinator User can only transmit a file.

** FACT Web Services ID is used only for automated submission of files. Web Services ID can not be combined with other FACT user access role(s) and must be requested separately from other roles.

(Please reference the Risk Sharing Pool Operational Manual for further details regarding the user types)

INFORMATION AND SYSTEM SECURITY STATEMENT

In consideration of Facility Association allowing my authorized access to its computer facilities, I understand and agree, that:

1. I will not disclose to any person my User ID(s) as assigned by Facility Association or my password(s);
2. I will use only ID(s) or password(s) assigned to me;
3. I will not pre-program any password(s) for automatic entry into any part of the computer facilities of Facility Association;
4. I will use the computer facilities of Facility Association and any software or other information relating to or contained in those facilities for the sole purpose of fulfilling my job duties;
5. I will treat as confidential any information of Facility Association and not disclose such information to any other party unless specifically authorized by Facility Association;
6. I will not, through the use of the computer facilities of Facility Association, infringe or violate the patent, copyright, license or proprietary right of any third party;
7. I will immediately advise Facility Association of any misuse of the computer and communication resources, the software or information relating to or contained in Facility Association facilities of which I become aware;
8. I will take all reasonable steps to prevent any pirated software, computer viruses and destructive programs from entering into any of the computer and communication resources of Facility Association;
9. I will not access, alter, destroy or copy any software or information relating to or stored in the computer facilities of the Facility Association unless specifically authorized by Facility Association;
10. I will take all reasonable steps to ensure the accuracy and completeness of the information I provide to Facility Association;
11. I will not collect, use or disclose personal information held by Facility Association unless specifically authorized by Facility Association.

I have read and I understand the security policies stated above and will comply with them. I understand that the failure to comply with the security policies may result in action by Facility Association.

IN WITNESS WHEREOF the undersigned has executed this Agreement on the ____ day of ____, 20__.

NAME: _____ **COMPANY:** _____

SIGNATURE: _____ **DATE:** _____

I have witnessed the signature of _____

NAME: _____ **COMPANY:** _____

SIGNATURE: _____ **DATE:** _____

Facility Association Correction and Transmission (FACT)

Guide for Data Submission via Web Service

Technical Documentation

April 2005

Table of Contents

PURPOSE3

BACKGROUND.....3

WEB SERVICES SETUP PROCESS3

FACT WEB SERVICE FOR FILE UPLOAD3

SAMPLE FACT WEB SERVICE CLIENT CODE.....3

Purpose

This document provides an overview of the setup process for the automated data submission through the FACT application. In addition it defines technical details for utilizing the web service function by describing the functionality and details how to connect to the web services at design time. It also contains a code sample for external user can to create testing programs.

Background

The newly introduced FA Alberta and New Brunswick Risk Sharing Pools data collection tool FACT includes 3 different methods for an insurer to submit data. The insurance company user can enter data by typing in a single transaction, submit a batch file of transactions through the FACT user interface or data can be submitted by web service. All validation will be the same whether data is entered manually or through a file. The following will describe the input and output for the web services so that external program will be able to utilize the service.

Note: Current Ontario RSP does not have this automated data submission feature. It will be available once the current PC pool package is replaced with the FACT application in late spring 2005. Testing of the FACT Ontario automated web service will be available in May 2005 and production submission are scheduled to start in June 1st, 2005.

Web Services Setup Process

Step 1 - Company Technical Evaluation

A company interested in the submission of RSP data through Web Service should first evaluate their internal capability to submit data in this method. Evaluation process should be conducted by the company technical development staff. To support their evaluation they should use this document.

Any technical support questions regarding the Web Service submission setup should be directed to IBC helpdesk at helpdesk@ibc.ca. Helpdesk will log the request and forward the question to the technical support staff. Any business related questions should be directed to Facility Association.

Step 2 - Obtaining the User ID

In order to submit data via the FACT automated submission a company requires a FACT Web Services User ID. The user id can be obtained by completing the FA Correction and Transmission (FACT) User ID request form and the Information and System Security Statement. Both forms are available from and must be submitted to FA. In the "User Access required for" section of FACT User ID form both PRODUCTION FACT Web Services and TEST FACT Web Services check boxes should be marked.

Once the FACT User ID request form has been submitted and validated by FA, an email, containing the system generated Web Service user-id and password, will be sent to the email address provided on the request form. The initial email will contain test user id, password and URL link for test region.

The emailed user will need to access FACT, via the URL provided in the email, to change the password. Once the original password has been changed the user id and new password must then be added to the member's web service code for access to FACT.

Step 3 - Testing

Throughout the FACT web services development and test preparation process any technical questions can be sent to IBC helpdesk at helpdesk@ibc.ca. Helpdesk will log the requests and then forward questions to the appropriate FACT technical support staff.

Submission of the first test file via web services should be coordinated with the FACT technical support staff. This can be done directly if there is a prior communication between the company and FACT support staff or through IBC helpdesk if it is the first time company is requesting web service support from IBC. The notification prior to sending the first test file is mandatory as the FACT Web Service test region availability has to be confirmed by the FACT support staff.

Note: It is important to use the proper user id, password and URL in testing the setup of the automated submission via web services. Once the submission setup has been tested the user will be sent a second email with Production user id, password and URL. The test and production user id's, passwords and URL links are all different. Once a company starts sending production data the test user id will be disabled.

Test URL: <https://uat.facilityassociation.com>

Step 4 - Production

Once the service has been successfully tested the next step is to prepare for the production submission. This preparation includes replacing the test user ID, password and URL within the automated submission web services scripts with those of production User ID, password and URL which will be provided following successful completion of testing. Again it would be beneficial if the transition from testing to production is coordinated with the FACT support staff at IBC so that they can monitor and provide support during this transition.

Production URL: <https://apps.facilityassociation.com>

Important

1. Any existing FACT user id set up for other user access can not be used for the automated data submission via web service.
2. Any company starting to submit RSP data via web service has to go through testing the transmission before starting with production submissions.
3. FACT Web Services user will not be able to access the FACT application in the manner as other FACT users who submit data manually. If FACT Web Services user id is used in a manual log on attempt the user will be presented with the message screen stating "Application is not available".
4. The production user id, password and URL link are all different from the test information and will be provided to the company once testing has been successfully completed. The password change procedure should be completed for both test user id and production user id and the reset passwords should then be used in the web services code.
5. A single completed FACT User ID request form is sufficient for both test and production setups if Production User Access and Test User Access areas are marked on the form.
6. A single FACT Web Services user id can be utilized to submit data for Alberta, New Brunswick or Ontario risk sharing pools. Within the submitted file input parameter code it will be identified for which province the data is being submitted. Mixing of data from different provinces into one file is not allowed.

FACT Web Service for File Upload

This web service will transfer a text file to IBC / FA server and invoke a preliminary check on the file record. The first 15 bytes are the key to identify an individual record. Its purpose is to maintain data integrity so that duplicate batches will not be entered into the system and to confirm that records are properly followed by a trailer record. In addition, the first byte indicates the records type (1 being Premium record, 2 being Premium trailer, 3 being Claim record and 4 is Claim trailer). Mixing of premium and claim records in a single file is not allowed. The current maximum length of record is set to 200 bytes. Anything longer than 200 bytes will be rejected. This may change dependent on the business requirement changes.

The following lists out the parameters for the web service:

Method Name: **UploadFileWebService**

Input Parameter: **string loginName** (user name created by FA to access this service),

string password (associated password for the user name),

string province 2 characters code:

 'AB' - Alberta,

 'NB' - New Brunswick

 'ON' - Ontario

byte [] fileContent (byte array for file content)

Output Parameter: **integer** (zero means successful completion)

Exception: SoapException will be thrown when there is an error

Sample FACT Web Service Client Code

Sample Console C# code

Step 1:

```
using System;
using System.Collections;
using System.Configuration;
using System.ComponentModel;
using System.Data;
using System.Web;
using System.Web.Services.Protocols;
using System.IO;

namespace TestWebServices
{
    /// <summary>
    /// Testing Class TestWS.
    /// </summary>
    class TestWS
    {
        /// <summary>
        /// Testing program for the web service
        /// </summary>
        [STAThread]
        static void Main(string[] args)
        {
            System.IO.FileStream fs = null;
            try
            {
                //
                // Add web reference to point to localhost
                // or other web service, then instantiate the instance
                //

                Fa.Fact.WebServices.UploadService srv
                    = new Fa.Fact.WebServices.UploadService();

                fs = new FileStream( "C:\\\\lw.txt", FileMode.Open,
                    FileAccess.Read );

                // Create a byte array of file stream length
                byte[] b = new byte[fs.Length - 1];

                //Read block of bytes from stream into the byte array
                fs.Read(b,0,System.Convert.ToInt32(fs.Length - 1));

                // invoke the service by providing the credential,
```

```
        // parameters & content of file pass in as byte array

        int rc = srv.UploadFileWebService
            ("FACTWS", "xxxxxx1", "AB", b);
        Console.WriteLine("Completed with return code : "+rc);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }
    finally
    {
        if (fs != null)
        {
            fs.Close();
        }
    }
}
}
```

Step 2:

Add web reference by pointing to

<http://uat.facilityassociation.com/Fa.Fact.WebServices/UploadService.asmx>

Or use the following files

A) ..\Fa.Fact.WebServices\uploadservice.wsdl

```
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns:s1="http://facilityassociation.com/AbstractTypes"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:s0="http://facilityassociation.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
targetNamespace="http://facilityassociation.com"
xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://facilityassociation.com">
      <s:element name="UploadFileWebService">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="loginName"
type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="password"
type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="province"
type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="fileContent"
type="s:base64Binary" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="UploadFileWebServiceResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1"
name="UploadFileWebServiceResult" type="s:int" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="UploadFile">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="loginName"
type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="password"
type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </types>

```

```

        <s:element minOccurs="1" maxOccurs="1" name="verify"
type="s:int" />
        <s:element minOccurs="0" maxOccurs="1" name="province"
type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="fileContent"
type="s:base64Binary" />
    </s:sequence>
</s:complexType>
</s:element>
<s:element name="UploadFileResponse">
    <s:complexType>
        <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="UploadFileResult"
type="s:int" />
        </s:sequence>
    </s:complexType>
</s:element>
<s:element name="int" type="s:int" />
</s:schema>
<s:schema
targetNamespace="http://facilityassociation.com/AbstractTypes">
    <s:complexType name="StringArray">
        <s:complexContent mixed="false">
            <s:restriction base="soapenc:Array">
                <s:sequence>
                    <s:element minOccurs="0" maxOccurs="unbounded" name="String"
type="s:string" />
                </s:sequence>
            </s:restriction>
        </s:complexContent>
    </s:complexType>
</s:schema>
</types>
<message name="UploadFileWebServiceSoapIn">
    <part name="parameters" element="s0:UploadFileWebService" />
</message>
<message name="UploadFileWebServiceSoapOut">
    <part name="parameters" element="s0:UploadFileWebServiceResponse" />
</message>
<message name="UploadFileSoapIn">
    <part name="parameters" element="s0:UploadFile" />
</message>
<message name="UploadFileSoapOut">
    <part name="parameters" element="s0:UploadFileResponse" />
</message>
<message name="UploadFileWebServiceHttpGetIn">
    <part name="loginName" type="s:string" />
    <part name="password" type="s:string" />
    <part name="province" type="s:string" />
    <part name="fileContent" type="s1:StringArray" />
</message>
<message name="UploadFileWebServiceHttpGetOut">
    <part name="Body" element="s0:int" />
</message>

```

```
<message name="UploadFileHttpGetIn">
  <part name="loginName" type="s:string" />
  <part name="password" type="s:string" />
  <part name="verify" type="s:string" />
  <part name="province" type="s:string" />
  <part name="fileContent" type="s1:StringArray" />
</message>
<message name="UploadFileHttpGetOut">
  <part name="Body" element="s0:int" />
</message>
<message name="UploadFileWebServiceHttpPostIn">
  <part name="loginName" type="s:string" />
  <part name="password" type="s:string" />
  <part name="province" type="s:string" />
  <part name="fileContent" type="s1:StringArray" />
</message>
<message name="UploadFileWebServiceHttpPostOut">
  <part name="Body" element="s0:int" />
</message>
<message name="UploadFileHttpPostIn">
  <part name="loginName" type="s:string" />
  <part name="password" type="s:string" />
  <part name="verify" type="s:string" />
  <part name="province" type="s:string" />
  <part name="fileContent" type="s1:StringArray" />
</message>
<message name="UploadFileHttpPostOut">
  <part name="Body" element="s0:int" />
</message>
<portType name="UploadServiceSoap">
  <operation name="UploadFileWebService">
    <input message="s0:UploadFileWebServiceSoapIn" />
    <output message="s0:UploadFileWebServiceSoapOut" />
  </operation>
  <operation name="UploadFile">
    <input message="s0:UploadFileSoapIn" />
    <output message="s0:UploadFileSoapOut" />
  </operation>
</portType>
<portType name="UploadServiceHttpGet">
  <operation name="UploadFileWebService">
    <input message="s0:UploadFileWebServiceHttpGetIn" />
    <output message="s0:UploadFileWebServiceHttpGetOut" />
  </operation>
  <operation name="UploadFile">
    <input message="s0:UploadFileHttpGetIn" />
    <output message="s0:UploadFileHttpGetOut" />
  </operation>
</portType>
<portType name="UploadServiceHttpPost">
  <operation name="UploadFileWebService">
    <input message="s0:UploadFileWebServiceHttpPostIn" />
    <output message="s0:UploadFileWebServiceHttpPostOut" />
  </operation>
</portType>
```

```

    <operation name="UploadFile">
      <input message="s0:UploadFileHttpPostIn" />
      <output message="s0:UploadFileHttpPostOut" />
    </operation>
  </portType>
  <binding name="UploadServiceSoap" type="s0:UploadServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
    <operation name="UploadFileWebService">
      <soap:operation
soapAction="http://facilityassociation.com/UploadFileWebService"
style="document" />
        <input>
          <soap:body use="literal" />
        </input>
        <output>
          <soap:body use="literal" />
        </output>
      </operation>
    <operation name="UploadFile">
      <soap:operation
soapAction="http://facilityassociation.com/UploadFile" style="document" />
        <input>
          <soap:body use="literal" />
        </input>
        <output>
          <soap:body use="literal" />
        </output>
      </operation>
    </binding>
  <binding name="UploadServiceHttpGet" type="s0:UploadServiceHttpGet">
    <http:binding verb="GET" />
    <operation name="UploadFileWebService">
      <http:operation location="/UploadFileWebService" />
      <input>
        <http:urlEncoded />
      </input>
      <output>
        <mime:mimeXml part="Body" />
      </output>
    </operation>
    <operation name="UploadFile">
      <http:operation location="/UploadFile" />
      <input>
        <http:urlEncoded />
      </input>
      <output>
        <mime:mimeXml part="Body" />
      </output>
    </operation>
  </binding>
  <binding name="UploadServiceHttpPost" type="s0:UploadServiceHttpPost">
    <http:binding verb="POST" />
    <operation name="UploadFileWebService">

```

```
<http:operation location="/UploadFileWebService" />
<input>
  <mime:content type="application/x-www-form-urlencoded" />
</input>
<output>
  <mime:mimeType part="Body" />
</output>
</operation>
<operation name="UploadFile">
  <http:operation location="/UploadFile" />
  <input>
    <mime:content type="application/x-www-form-urlencoded" />
  </input>
  <output>
    <mime:mimeType part="Body" />
  </output>
</operation>
</binding>
<service name="UploadService">
  <port name="UploadServiceSoap" binding="s0:UploadServiceSoap">
    <soap:address
location="http://localhost/Fa.Fact.WebServices/uploadservice.asmx" />
    </port>
    <port name="UploadServiceHttpGet" binding="s0:UploadServiceHttpGet">
    <http:address
location="http://localhost/Fa.Fact.WebServices/uploadservice.asmx" />
    </port>
    <port name="UploadServiceHttpPost" binding="s0:UploadServiceHttpPost">
    <http:address
location="http://localhost/Fa.Fact.WebServices/uploadservice.asmx" />
    </port>
  </service>
</definitions>
```

B) ..\Fa.Fact.WebServices\ uploadservice.disco

```
<?xml version="1.0" encoding="utf-8"?>
<discovery xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://schemas.xmlsoap.org/disco/">
  <contractRef
ref="http://localhost/Fa.Fact.WebServices/uploadservice.asmx?wsdl"
docRef="http://localhost/Fa.Fact.WebServices/uploadservice.asmx"
xmlns="http://schemas.xmlsoap.org/disco/scl/" />
  <soap address="http://localhost/Fa.Fact.WebServices/uploadservice.asmx"
xmlns:q1="http://facilityassociation.com" binding="q1:UploadServiceSoap"
xmlns="http://schemas.xmlsoap.org/disco/soap/" />
</discovery>
```

C) ..\Fa.Fact.WebServices\Reference.map

```
<?xml version="1.0" encoding="utf-8"?>
<DiscoveryClientResultsFile xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Results>
    <DiscoveryClientResult
referenceType="System.Web.Services.Discovery.ContractReference"
url="http://localhost/Fa.Fact.WebServices/uploadservice.asmx?wsdl"
filename="uploadservice.wsdl" />
    <DiscoveryClientResult
referenceType="System.Web.Services.Discovery.DiscoveryDocumentReference"
url="http://localhost/Fa.Fact.WebServices/uploadservice.asmx?disco"
filename="uploadservice.disco" />
  </Results>
</DiscoveryClientResultsFile>
```

D) ..\Fa.Fact.WebServices\Reference.cs

```

i»¿//-----
-----
// <autogenerated>
//     This code was generated by a tool.
//     Runtime Version: 1.0.3705.288
//
//     Changes to this file may cause incorrect behaviour and will be lost
if
//     the code is regenerated.
// </autogenerated>
//-----
-----

//
// This source code was auto-generated by Microsoft.VSDesigner, Version
1.0.3705.288.
//
namespace TestWebServices.Fa.Fact.WebServices {
    using System.Diagnostics;
    using System.Xml.Serialization;
    using System;
    using System.Web.Services.Protocols;
    using System.ComponentModel;
    using System.Web.Services;

    /// <remarks/>
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]

    [System.Web.Services.WebServiceBindingAttribute(Name="UploadServiceSoap",
    Namespace="http://facilityassociation.com")]
    public class UploadService :
    System.Web.Services.Protocols.SoapHttpClientProtocol {

        /// <remarks/>
        public UploadService() {
            this.Url =
"http://localhost/Fa.Fact.WebServices/uploadservice.asmx";
        }

        /// <remarks/>

        [System.Web.Services.Protocols.SoapDocumentMethodAttribute("http://facility
association.com/UploadFileWebService",
RequestNamespace="http://facilityassociation.com",
ResponseNamespace="http://facilityassociation.com",
Use=System.Web.Services.Description.SoapBindingUse.Literal,
ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)]
        public int UploadFileWebService(string loginName, string password,
string province,

```

```

[System.Xml.Serialization.XmlElementAttribute(DataType="base64Binary")]
System.Byte[] fileContent) {
    object[] results = this.Invoke("UploadFileWebService", new
object[] {
        loginName,
        password,
        province,
        fileContent});
    return ((int)(results[0]));
}

/// <remarks/>
public System.IAsyncResult BeginUploadFileWebService(string
loginName, string password, string province, System.Byte[] fileContent,
System.AsyncCallback callback, object asyncState) {
    return this.BeginInvoke("UploadFileWebService", new object[] {
        loginName,
        password,
        province,
        fileContent}, callback, asyncState);
}

/// <remarks/>
public int EndUploadFileWebService(System.IAsyncResult asyncResult)
{
    object[] results = this.EndInvoke(asyncResult);
    return ((int)(results[0]));
}

/// <remarks/>

[System.Web.Services.Protocols.SoapDocumentMethodAttribute("http://facility
association.com/UploadFile",
RequestNamespace="http://facilityassociation.com",
ResponseNamespace="http://facilityassociation.com",
Use=System.Web.Services.Description.SoapBindingUse.Literal,
ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)]
public int UploadFile(string loginName, string password, int
verify, string province,
[System.Xml.Serialization.XmlElementAttribute(DataType="base64Binary")]
System.Byte[] fileContent) {
    object[] results = this.Invoke("UploadFile", new object[] {
        loginName,
        password,
        verify,
        province,
        fileContent});
    return ((int)(results[0]));
}

/// <remarks/>
public System.IAsyncResult BeginUploadFile(string loginName, string
password, int verify, string province, System.Byte[] fileContent,
System.AsyncCallback callback, object asyncState) {

```

```
        return this.BeginInvoke("UploadFile", new object[] {
            loginName,
            password,
            verify,
            province,
            fileContent}, callback, asyncState);
    }

    /// <remarks/>
    public int EndUploadFile(System.IAsyncResult asyncResult) {
        object[] results = this.EndInvoke(asyncResult);
        return ((int)(results[0]));
    }
}
```