The Modeling of Communication with Other Systems in Process Automation Applications

Dhori Terpo
University “E. Çabej”, Faculty of Natural Sciences
Department of Mathematics & Informatics, Gjirokastra, Albania

Endrit Xhina
University of Tirana, Faculty of Natural Sciences
Department of Informatics, Tirana, Albania

Ina Hyseni
University of Tirana, Faculty of Natural Sciences
Department of Informatics, Tirana, Albania

Endri Bimbari
University of Tirana, Faculty of Natural Sciences
Department of Informatics, Tirana, Albania

Copyright © 2015 Dhori Terpo, Endrit Xhina, Ina Hyseni and Endri Bimbari. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

Usage of process automation applications in public institutions as well as in the business area implies the existing of communication facilities with other systems. The software should have facilities which allow to exchange data with other systems as a response of users’ or system’s events. Tools and techniques are available to enable this functionality, among which the web services. This paper will present a web services based solution for exchanging data with other systems built for a process automation application. An illustrative example of usage of this solution in a scenario which involves the communication between e-government systems will also be presented in this paper.
Keywords: Web Services, Process Automation Applications, e-Government

1 Web Services in .NET

The World Wide Web Consortium (W3C) definition states “a Web Service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web Service in a manner prescribed by its definition, using XML-based messages conveyed by internet protocols”. [1]

Figure 1 shows the basic Web Services architecture [2].

Figure 1. The basic Web Services architecture

The Web Services architecture is based upon the interactions between three roles:

- Service requestor - initiates the execution of service
- Service provider - processes a Web service request
- Discovery agency - an optional intermediate broker through which a Web service description is published and made discoverable.

The operations supported by the roles are:

- A service provider publishes services to the discovery agency.
- A service requestor finds services in the discovery agency.
- Service requestor and provider interact. This logical operation consists of three steps-locate, contact and invoke.
Web services are built on Web technologies. The three core technologies that form the foundation for Web services are the Hypertext Transfer Protocol (HTTP), the Extensible Markup Language (XML), and the Simple Object Access Protocol (SOAP).

HTTP is the protocol widely used by web services to send and receive messages. XML is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable. SOAP is a simple XML-based protocol to let applications exchange information over HTTP.

The easiest way to create Web Services in .Net is to build them using ASP.NET. A basic example of a web service that exposes a method that returns the string “Hello Web Service” to the client is shown in Listing 1.

Listing 1. Example of a web service that returns the string “Hello Web Service”

```csharp
using System.Web.Services;
namespace WebServiceDemo
{
    public class Service : WebService
    {
        [WebMethod(Description = "Returns the string “Hello Web Service”")]
        public string HelloWS()
        {
            return "Hello Web Service";
        }
    }
}
```

2 Messaging Models Used in Web Services

The two Web Service messaging models are:\n
- **Synchronous Model**
  When the client application make a synchronous call to a Web service method, the client application will not process any other commands until the SOAP response comes back from the server.

- **Asynchronous Model**
  When the client application make an asynchronous call to a Web service method, the client sends the SOAP request, but it is not blocked waiting for the SOAP response. The Web service does not require any special configuration to support asynchronous calls. In fact, as far as the server knows, the call is synchronous. All the additional processing is done at the client.
An example of invoking a Web Service method synchronously is shown in Listing 2 and invoking a Web service method asynchronously is shown in Listing 3.

**Listing 2. Invoking a Web service method synchronously**

```csharp
using System;
namespace WebServiceClient
{
    public partial class WebClient : System.Web.UI.Page
    {
        protected void btnSync_Click(object sender, EventArgs e)
        {
            localhost.Service client = new localhost.Service();
            TextBox1.Text = client.HelloWS();
        }
    }
}
```

**Listing 3. Invoking a Web service method asynchronously**

```csharp
using System;
namespace WebServiceClient
{
    public partial class WebClient : System.Web.UI.Page
    {
        protected void btnSync_Click(object sender, EventArgs e)
        {
            localhost.Service client = new localhost.Service();
            TextBox1.Text = client.HelloWS();
        }

        protected void btnAsynch_Click(object sender, EventArgs e)
        {
            localhost.Service client = new localhost.Service();
            client.HelloWSCompleted +=
                new localhost.HelloWSCompletedEventHandler(HelloWSCompleted);
            client.HelloWSAsync();
        }

        void HelloWSCompleted(object sender, localhost.HelloWSCompletedEventArgs e)
        {
            TextBox2.Text = e.Result;
        }
    }
}
```
3 Authentication in Web services

Authentication refers to verifying that the identity of an entity is in fact that which it claims to be. The entity trying to have its identity authenticated is known as the principal. The evidence used to prove the principal’s identity is known as the credentials. If the correct credentials are used, the principal is assumed to be who it claims to be.

A variety of technologies and approaches can be used to implement authentication for Web services. These approaches can generally be classified as system-level approaches, application-level approach, or third-party approach.

**System-level approach** do not require custom application (or Web service) programming to implement. Nor does it require any changes to the Web service if the authentication approach is changed. Usually, the operating system or the Web server handles authentication prior to forwarding the SOAP request to the Web service. Common system-level approaches to authentication include basic password, encrypted passwords, and digital certificates.

**Application-level approaches** to authentication require custom development, and usually have to be modified with changes to the authentication mechanism. Application-level authentication approaches can pass credentials as part of the SOAP message. In this case, the Web service must parse the credentials as well as implement authentication mechanisms itself. SOAP on top of HTTP exposes the credentials as clear text, and facilitates misappropriation. SSL can be used to encrypt the data for all SOAP messages sent to the other operation of the Web service.

**Third-party authentication** services may also be available. Single sign-on capabilities are of particular interest to Web service environments, which are comprised of a large number of heterogeneous services, each of which may use different authentication mechanism. With a single sign-on scheme, service requesters need only maintain a single credential. The third-party single sign-on service manages and maps the single credential held by service requesters to each of the service providers. The complexities of maintaining, managing, and revoking authentication credentials are handle by third-party service provider.

4 Web service metadata query

Web services use metadata to describe what other endpoints need to know to interact with them. **Web service Description Language (WSDL)** is an XML-based format file for describing Web services. Through the WSDL, a Web services client learns where a service can be accessed, what operations the service performs, the communication protocols the service supports, and the correct format for sending messages to the service.
A WSDL file is an XML document that describes a Web service using six main elements:

- Port type – groups and describes the operations performed by the service through the defined interface.
- Port – specifies an address for a binding, i.e., defines a communication port.
- Message – describes the names and format of the messages supported by the service.
- Types – defines the data types (as defined in an XML Schema) used by the service for sending messages between the client and server.
- Binding – defines the communication protocols supported by the operations provided by the service.
- Service – specifies the address (URL) for accessing the service.

WSDL is often used in combination with SOAP and an XML Schema to provide Web services over the Internet. A client program connecting to a Web service can read the WSDL file to determine what operations are available on the server. Any special data types used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP messages to actually call one of the operations listed in the WSDL file.

Listing 4 presents an example of reading the wsdl file that describes the web service of Listing 1. The operations of the web service are shown in Figure 2.

**Listing 4. Displaying the operations of a web service**

```csharp
using System;
using System.Text;
using System.Web.Services.Description;
using System.Xml;
using System.Xml.Schema;
using System.Xml.Serialization;

namespace WebService
{
    public partial class WebForm2 : System.Web.UI.Page
    {
        ServiceDescription serviceDescription;
        BindingCollection bindColl;
        PortTypeCollection portTypColl;
        MessageCollection msgColl;
        Types types;
        XmlSchemas schemas;
        String output = string.Empty;
        StringBuilder sb;

        protected void Button1_Click(object sender, EventArgs e)
```
{ 
    Label2.Text = "The following operations are supported";
    XmlReader reader =
        new XmlTextReader(TextBox1.Text + "?WSDL");

    serviceDescription = ServiceDescription.Read(reader);
    bindColl = serviceDescription.Bindings;
    portTypColl = serviceDescription.PortTypes;
    msgColl = serviceDescription.Messages;
    types = serviceDescription.Types;
    schemas = types.Schemas;

    foreach (Service service in serviceDescription.Services)
    {
        Port port = service.Ports[0];
        string binding = port.Binding.Name;
        Binding bind = bindColl[binding];
        PortType portType = portTypColl[bind.Type.Name];
        sb = new StringBuilder();
        sb.Append("<ul>");
        foreach (Operation operation in portType.Operations)
        {
            string operationName = operation.Name;
            string operationDocumentation = operation.Documentation;
            OperationMessageCollection opMsgColl = operation.Messages;
            OperationInput opInput = opMsgColl.Input;
            OperationOutput opOutput = opMsgColl.Output;
            string inputMsg = opInput.Message.Name;
            string outputMsg = opOutput.Message.Name;
            Message msgOutput = msgColl[outputMsg];
            sb.Append("<li>");
            DisplayParametersAndReturnType(msgOutput, "output");
            sb.Append(operationName.ToString());
            sb.Append("&nbsp");
            sb.Append("(");
            Message msgInput = msgColl[inputMsg];
            DisplayParametersAndReturnType(msgInput, "input");
            sb.Append("")
            sb.Append("</br>");
            sb.Append(operationDocumentation);
            sb.Append("</li>");
        }
        sb.Append("</ul>");
    }
    Label3.Text = sb.ToString();
}
public void DisplayParametersAndReturnType(Message msg,
    string parameter)
{
    foreach (MessagePart msgpart in msg.Parts)
    {
        XmlQualifiedName typename = msgpart.Element;
        XmlSchemaElement lookup = (XmlSchemaElement)
            schemas.Find(typename, typeof(XmlSchemaElement));
        if ((msgpart.Name == "parameters") | (msgpart.Name == "Body"))
        {
            XmlSchemaComplexType complexType =
                (XmlSchemaComplexType)lookup.SchemaType;
            XmlSchemaSequence sequence =
                (XmlSchemaSequence)complexType.Particle;
            if (sequence != null)
            {
                foreach (XmlSchemaElement childElement in sequence_Items)
                {
                    sb.Append(childElement.SchemaTypeName.Name);
                    sb.Append("&nbsp");
                    if (parameter == "input")
                    {
                        sb.Append(childElement.Name);
                    }
                    sb.Append("&nbsp");
                }
            }
        }
    }
}

The following operations are supported:

- string HelloWS()
  Returns the string "Hello World"

Figure 2. Displaying the operations of a web service
5 Case Study

Providing e-government services at Local Government Units on behalf of central government institutions

The case of National Registration Center (NRC) in Albania

National Registration Center in Albania is a central government institution which serves as a single point of service for business registration and related services. In order to bring the service near the business community the law for NRC ¹ foresees the service provisioning on NRC offices as well as in dedicated service desks within Local Government Units. NRC has opened service desks in municipalities and some big communes as an application of the requirement set by the law of NRC.

Within the OSSh office² of local government units it is offered the service of NRC through the means of a dedicated desk with a PC, connected to the system of NRC. This computer is connected through VPN connection to the system of NRC and uses NRC software to communicate with the back end. A sample list of services offered through these service desks is listed below:

- Limited Company Registration
- Joint-Stock Company Registration
- Legal Person Registration
- Natural Person Registration
- Legal Person Suspension
- Natural Person Suspension
- Natural Person Deregistration

Changes to the legal base in Albania, change the organization of local government units through the creation of bigger units as an amalgamation of municipalities

¹ Law No. 9723, date 3.5.2007, “On National Registration Center”

² OSSh Office = One Stop Shop Office, a single point of services for all administrative services
and communes with a municipality serving as the center of the local government unit. Communes and any other eventual municipality part of the amalgamation will become OSSh model based service points.

This structural change creates bigger units and creates an opportunity to expand the NRC services to all service points within all the service points of the new local government unit.

Our proposal is to use the Process Automation Software which is already in use in One Stop Shops to serve citizens as a middle layer between the operator and the NRC system instead of having a dedicated computer connected to NRC system. Smart Processes\(^3\) would use its interface configuration tool called form designer\(^6\), to design the interface which will allow the operator to enter the request of the citizen and prepare the response. Then through the “web service control button”, the citizen’s request will be sent to NCR system. If the request involves querying NRC system for data then the result of the query will populate the already designed form inputs and the operator will be able to print the standard document requested by the citizen.

If the request involves case processing on the NRC system then Smart Processes will schedule a periodic call until it gets a final status of the case from the NRC system. Once the final status case is received by the NRC system an additional web service query is done by the Smart Processes in order to get the data expected for the specific case. The data received populates the form inputs and the case status in Smart Processes is changed. The notification subsystem part of the Smart Processes send a case status change notification message to the operator and the citizen using a selected communication channel either email or SMS.

Figure 3 illustrates the architectural model of the proposed system.

---

\(^3\) Smart Processes is the process automation software used for the presented solution.
6 Conclusions

Service provisioning either in government or business is a dynamic field which implies changes in the data inputs, workflow and communication. Changes might caused by many factors being those legal, regulatory rules, opportunities, service improvement, etc. Usage of Process Automation Software in these scenarios enables institutions to quickly adopt to change. An essential part of the Process Automation Software is the ability to communicate and exchange data with other systems. This paper proposed Web services as a standard way of communication between systems.

The implementation of the web services within a Process Automation Software called Smart Processes is presented afterwards in the paper. Finally a scenario of communication between two government institutions, one central and one local is modeled using Smart Processes as a middle layer between the service operators at the local government with the information system at the central government institution.

References


Received: March 10, 2015; Published: March 24, 2015