7-12-2010

Web Service for Knowledge Management Information Tool (KMIT) Hotline module and its Security

Harini Kondamudi
Florida International University, hkond001@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

WEB SERVICE FOR KNOWLEDGE MANAGEMENT INFORMATION TOOL (KMIT) HOTLINE MODULE AND ITS SECURITY

A thesis submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Harini Kondamudi

2010

To: Dean Amir Mirmiran
College of Engineering and Computing

This thesis, written by Harini Kondamudi, and entitled Web Service for Knowledge Management Information Tool (KMIT) Hotline module and its Security, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this thesis and recommend that it be approved.

Leonel E Lagos

Peter J Clarke

Ming Zhao, Major Professor

Date of Defense: July 12, 2010

The thesis of Harini Kondamudi is approved.

Dean Amir Mirmiran
College of Engineering and Computing

Interim Dean Kevin O'Shea
University Graduate School

Florida International University, 2010

DEDICATION

I dedicate this dissertation to my dad Satya and my fiancé Dilip. Without their patience, understanding, support, and most of all love, the completion of this work would not have been possible.

ACKNOWLEDGMENTS

I wish to thank the members of my committee, Dr. Ming Zhao, Dr. Leonel E Lagos and Dr. Peter J Clarke for their help and support. I want to specially thank Mr. Himanshu Upadhyay, my mentor at Applied Research Center (FIU – ARC) without whose guidance my research is incomplete. His gentle but firm direction has been most appreciated. My fellow graduate student and friend, Gowthami Thota was very helpful in guiding me all through the thesis. My fellow graduate student and friend, Ravi Prasanth Gudavalli was extremely patient to work with the formatting of my document. My colleague Peggy Shoffner who reviewed the thesis and gave her valuable comments. I immensely thank you all for your support.

ABSTRACT OF THE THESIS

WEB SERVICE FOR KNOWLEDGE MANAGEMENT INFORMATION TOOL (KMIT) HOTLINE MODULE AND ITS SECURITY

by

Harini Kondamudi

Florida International University, 2010

Miami, Florida

Professor Ming Zhao, Major Professor

This thesis presents the development of a Web Service for the Hotline module of the Knowledge Management Information Tool (KMIT), a tool that is custom built for the decontamination & decommissionin (D&D) community of the Department Of Energy (DOE). The Hotline module allows interested users to post problems to specific areas of interest in the field of D&D. Various clients working with DOE and KMIT want to display the latest published problems of KMIT Hotline search in their own applications on a regular basis. Considering one of the major benefits of Web Services is the ease of integration of one piece of software with another, the Hotline Service is successfully developed and can be plugged into client's applications by adding a reference to it. In such a distributed environment, messages can flow from node to node, through firewalls, onto the internet and through various intermediaries. This introduces a variety of message security threats. The research for this thesis included a study of the various security risks and scenarios. Appropriate security model is designed and is successfully implemented. Hotline Service can authenticate the client and ensure confidentiality making the service secure to communicate with.

TABLE OF CONTENTS

CHAPTER PAGE

CHAPTER PAGE

LIST OF FIGURES

FIGURE PAGE

1.0 INTRODUCTION:

Application integration is one of the most important issues that is currently being faced by information systems. Application integration is the mechanism that enables different software systems to share, and use information in a convenient way [1]. The accomplishment of application integration can be made easier if every business function written became available to another new application by simply adding a reference to it, and if all these functions could be discovered and used at runtime. However, organizations develop different application systems using different technologies and this makes the process of communication between the applications tedious. Also, many third party applications are not designed to communicate with other applications and it requires a lot of time and effort to enable data integration among these applications. *XML Web Services* are one solution for application integration.

KMIT is a Knowledge Management Information Tool that is custom built for the D&D community. This system is being developed by Florida International University- Applied Research Center in collaboration with the Department of Energy (DOE EM 20), the Energy Facility Contractors Group (EFCOG) and the ALARA centers at the Hanford and Savannah River sites. The Hotline Module of KMIT allows interested users to post questions/problems related to a specific area of interest in the area of decontamination and decommissioning (D&D). Various clients working with DOE and KMIT who want to display the latest published problems of KMIT hotline search in their own applications on a regular basis. Considering one of the major benefits of Web Services is the ease of integration of one piece of software with another, we proposed the idea of developing a Web Service for the KMIT Hotline module which can be plugged into other client's applications. With their greater amount of use and ease of integration, they are open to serious attacks. The purpose of this study is to secure, the service which is always a top issue. Even though there are various security mechanisms available, there is little knowledge on which model to apply for a specific scenario. The challenge lies in testing the service to know if it actually implements the specific security model. The proposed idea to achieve this is to intercept the Simple Object Access Protocol (SOAP) messages being transferred between the client and the service.

In this thesis, I present the various methods of security mechanisms available to protect Web Services and the specific security model designed to secure KMIT HOTLINE WEB SERVICE. The security model follows WS- SECURITY protocol. The security concept generally addresses issues related to authentication, integrity and confidentiality. The study here is particularly about authenticating the client applications and allowing them to access the service. Study about integrity and confidentiality are left for future work. This model is the outcome of the study of various authentication standards. This is accomplished by studying the authentication mechanisms available, and picking the right method to fit the Hotline Service security requirements and the scenario. The security model follows the implementation of 'message security' specified by Microsoft Developer Network (MSDN) [20] using Windows Communication Foundation WCF.

I worked in a team of four with my advisor, Prof. Ming Zhao; mentor, Mr. Himanshu Upadhyay; and a graduate student, Gowthami Thota. My role in the team is to research the various scenarios and security standards, develop the HOTLINE WEBSERVICE, add the elements of authentication and test if the authentication method of security was properly implemented. The HOTILINE WEB SERVICE development is divided into three phases. The first phase is to build a platform independent Application Program Interface (API) to expose the endpoints of the service. The Hotline Service consists of one end point which opens the gateway for communication to the clients. The second phase is to host the service on Internet Information Service (IIS). For a service to be active, it must be hosted within a run-time environment that creates it and controls its context and lifetime. The IIS hosting option in WCF is integrated with ASP.NET and uses the features these technologies offer, such as process recycling, idle shutdown, process health monitoring, and message-based activation [2]. The third phase is to write client applications to test if the service is running. WCF provides a versatile and interoperable platform for exchanging secure messages based upon the existing security infrastructure and the recognized security standards for SOAP messages.

The security model describes the WS-Security protocol and how WCF integrates to use the protocol by using specific bindings. Also discussed are the available authentication models for security and why I chose the specific security model (*message security with username token and certificate authentication*) for KMIT. Further sections describe how the bindings are configured, how the certificates are created, signed and are used by the service and client for mutual authentication.

In sections 10.1 and 10.2, I describe the Fiddler, a debugging proxy, to verify if security is implemented properly to the service. Fiddler is worth studying as it acts as a Web debugging proxy which intercepts the SOAP messages and logs all HTTP(S) Hyper Text Transfer Protocol – Secured [HTTP(S)] traffic between the computer and the internet. Fiddler allows the inspection of all HTTP(S) traffic, the setting of breakpoints, and “fiddling" with incoming or outgoing data. Fiddler includes a powerful event-based scripting subsystem, and can be extended using any .NET language. Appropriate screenshots display how the fiddler intercepted the messages and how the authentication is verified. The results are also displayed verifying the usage of bit-256 encryption algorithm, the encrypted cipher text, binary token for exchange and the nonce values. Finally, I draw conclusions that the specific security model follows the standards of WS- SECURITY protocol, and validates them through the results displayed by fiddler.

The next chapter provides the background material, the history behind the evolution of web services and some important definitions used throughout the document. Chapter 3 describes the latest technology, Windows Communication Foundation, the advanced features of WCF and how the security can be implemented easily using the WCF framework. Chapter 4 discusses our motivation to the idea for development of Web Services, and chapter 5 describes the WS-Security protocol in detail, while chapter 6 describes why the developed services should be secured, and discusses various implementation techniques. Chapter 6 describes the first phase of implementation of the Hotline Web Service using WCF, right from describing the basics of WCF. Chapter 7 elaborates on Security model implemented for the Hotline Service and describes how the message security with username authentication is implemented using the digital certificates. Various sections of this chapter detail the procedure of adding and installing certificates to verify the incoming messages. Chapter 8 describes the second phase of development, i.e. to host the Hotline Service on IIS. Chapter 9 the third phase, explains the creation of a client to know if the service is responding to client requests. Chapter 10 explains the Fiddler tool. Appropriate results are documented which verify the security model implemented. Experimental results are shown in sections 10.1 and 10.2 which explain that security is successfully implemented. I then discuss the potential future work for Hotline Web Service, and give concluding remarks followed by a list of references.

2.0 BACKGROUND:

This chapter introduces some of the important definitions related to this research, as well as the terminology that will be used throughout the Thesis. Also discussed is the foundation to the study, Web Services, and why there is a need to develop a Web Service for Hotline module of KMIT, and specific details about the KMIT and Hotline module. More details are presented about the mechanisms available for communication before Web Services were developed. Even though later sections discuss the security in detail, a brief overview of it this presented.

2.1 Definitions:

Web Service	A Web Service is a software component that is described via Web Service Description Language (WSDL) and is capable of being accessed via standard network protocols such as but not limited to SOAP over HTTP [3]
XML	Extensible Markup Language, abbreviated describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language [ISO 8879] [3]
SOAP	SOAP, originally defined as Simple Object Access Protocol, is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment [4]

UDDI	The Universal Description, Discovery, and Integration (UDDI) specification defines a SOAP-based Web service for locating Web services and programmable resources on a network [5]
WSDL	WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.[6]
IIS	Internet Information Server. This is a Microsoft application that allows the creation of web-based applications that interact with COM server objects. Web pages that interact with COM server objects are called ASP pages [7]
KMIT	Knowledge Management Information Tool [8]
SOA	Service Oriented Architecture

2.2 Evolution of Web Services

Object Oriented (OO) programming joined the mainstream in the early 80's. Many saw it as the solution to the software crisis that resulted from the increasing complexity and size of the software being built. Most projects were late and over budget, and the end result was often unreliable. The promise of object orientation was that by structuring the code into objects that map to other objects in the solution domain, one would get code that was reusable and easily maintained. OO programming has improved software quality, but software projects are still often over budget and late. The 1990's saw the birth of component technology. Visual Basic is now ten years old, but it was revolutionary in many ways. It allows developers to build windows applications by dragging controls on to the form. In 1995, people talked about component technology and how it would make it possible to build applications by assembling components. Component re-use has turned out to be commercially very successful, but third party business components, have not lived up to its promise. Alongside, the internet has clearly had a major impact on society and our industry in particular.

In many ways, Web Services seemed to be an extension of component model to the internet, as essentially a Web Service is an application logic that can be used over the internet. Many of the promises of Web Services are the same as component technology, and they will allow us to assemble applications from pre-built application logic available somewhere on the internet. Similarly, Web Services will solve many problems we encounter when trying to build re-usable application logic and building applications that span the internet. In this chapter we will look at the "why", "what" and "how" of Web Services.

2.2.1 Why we need Web Services?

Web Services are interesting from several perspectives. From a technological perspective, Web Services try to solve some problems faced when using tightly-coupled technologies such as CORBA and DCOM. These are problems such as getting through firewalls, the complexities of the protocols, and integrating heterogeneous platforms. They are also interesting from an organizational or economic perspective, as they open up doors for new ways of doing business and dealing with organizational issues. Let us look at some reasons why we need services

- Web Services are loosely coupled with clients. A client makes a request to a service. The service returns the result and the connection is closed. There is no permanent connection, and none of the complexities.
- The Web Service may extend its interface, add new methods and parameters without affecting the clients, as long as it still services the old methods and clients.
- Web Services are stateless. They do not hold on to a state on behalf of the client. This makes scalability easier.
- The foundation of Web Services- SOAP – Simple Object Access Protocol is very easy to implement compared to DCOM and CORBA
- For serialization, DCOM and CORBA are based on complex formats. The serialization of Web Services is based on XML and XML schema specification. XML is simple, extensible and readable.
- Where DCOM and CORBA use IDL to describe interface, Web Services use WSDL to describe its interface which is more flexible.

2.2.2 What are Web Services?

The term Web Service opens doors for confusion during early stages. Is a hosted solution provided through the internet a Web Service? Then, what about software that is downloaded from the internet when needed? Is that a Web Service too?

The following defines a Web Service **"A web service is an application logic that is accessible to a program via standard web protocols in a platform – independent way."**

If we break up the definition,

- **Application Logic:** A Web Service exposes some application logic or code. This code can do anything a program can do, look up databases or doing calculations etc.
- **Accessible to program:** Where most websites today are accessed by humans, Web Services are accessed by computer programs.
- **Standard Web Protocols:** The whole concept of Web Services is based on a set of standards as HTTP, XML, SOAP, WSDL, and UDDI.
- **Platform Independent:** Web Services can be implemented on any platform. The standard protocols are not proprietary to any vendor, and are supported by all major vendors.

2.2.3 How to implement Web Services with ASP.NET

After talking about why and what they are, this chapter discusses how Web Services are implemented.

Web Services are based on SOAP, and SOAP is independent of how the services are implemented. There are many ways to implement services. All vendor developer tools are or will be providing tools to develop services. IBM has Web Services toolkit, Apache SOAP protocol has a tool kit and many more. Even when committed to Microsoft, there are several ways to implement SOAP based Web Services, including:

- Hand coding the service and formatting the SOAP and XML by hand. This is a very tedious option.
- Use SOAP toolkit, downloadable from http://msdn.microsoft.com
- Use ATL server. ATL is a part of Visual studio.net
- Use .NET remoting which allows classes inherting from a base class called MarshalByRefObject to be exposed as Web Services using SOAP.
- Use ASP.NET

From the research and my opinion the .NET framework is the superior platform for building, deploying and consuming Web Services. Other platforms and tools have Web Services bolted on top of them. The .NET framework is built from the ground to support XML and Web Services.

2.3 Service oriented architecture

It would be easy to conclude that the move to Service orientation really commenced with Web Services about three years ago. However, Web Services were merely a step along a much longer road. What's important to recognize is that Web Services are part of the wider picture that is SOA. Web Services provide us with certain architectural characteristics and benefits, specifically, platform independence, loose coupling, self description, and discovery and they can enable a formal separation between the provider and consumer because of the formality of the interface.

Service is the important concept. Web Services are the set of protocols by which Services can be published, discovered and used in a technology neutral, standard form [9]. It seems probable that eventually most software capabilities will be delivered and consumed as services. Over time, the level of abstraction at which functionality is specified, published and or consumed has gradually become higher and higher. Technology has progressed from modules, to objects, to components, and now to services. A distributed system consists of diverse, discrete software agents that must work together to perform some tasks. Furthermore, the agents in a distributed system do not operate in the same processing environment, so they must communicate by hardware/software protocol stacks over a network. This means that communications with a distributed system are intrinsically less fast and reliable than those using direct code invocation and shared memory. [10]. Many organizations currently offer service which implements a general purpose API and can provide basic create, read, update and delete (CRUD) access to the database through web service. While we see that there is nothing wrong with this kind of implementation, it is also essential that users understand the underlying model and comply with the business rules of the organization. The WSDL tells nothing about the business or entities. This could be an example of Web Services without SOA. *SOA is not just an architecture of services seen from a technology perspective, but the policies, practices, and frameworks by which we ensure the right services are provided and consumed* [9] .So, we need a frame work that can allow us to develop a good service that can provide:

- Interface related principles - technology neutrality, standardization and consumability.
- Design principles - these are more about achieving quality services, meeting real business needs, and making services easy to use, inherently adaptable, and easy to manage.

Here, the point to be noted is that, not all services should implement these qualities, however, by understanding the architecture, it can be noted that for a service like the KMIT Hotline which is to be used by multiple clients, the specification needs to be generalized, service should be abstract from implementation, and the developers of client applications should be unaware of the underlying model. The information provided to the client should be precise and the service should be offered at the required level of granularity.

There are certain things that are taken into consideration while designing a Web Service for KMIT which follows service oriented architecture

- The current user requirements - the key driving force for using Web Services is to ensure seamless end-to-end business processes. Thus, the development of a Hotline Service should follow the property of designing the service to accommodate the future needs of consumers
- For the consumers, the process must be organized in such a way that only the interface matters. They should have no dependence on how the service is being implemented. All they need to know is the location of the Hotline Service, what it does, and how it can be used. Because interface is the only thing for the consumer to interact with, considerable flexibility can be achieved.

- Similarly, the provider of the Hotline Service may have a different set of concerns, but the designer needs to develop and deliver a service that the consumer can use in any application. This shows that the focus of attention should be given to the interface – the description and contract.

2.4 Web Services with asp.net

Though there are several ways available to develop Web Services, the two major technologies that rule the business world applications are J2EE and .NET. This section compares the Web Service implementation the two major technologies J2EE and .NET, and the reason of choosing .NET over J2EE for KMIT. The comparison is done relating to the features present in each platform, the tools and the resources offered by the two and compatibility with the rest.

2.4.1 In J2EE

J2EE is a set of specifications created by the java community process (JCP). It is used for developing enterprise level applications. As a framework for the development of multitier enterprise applications, it makes the job of a developer easy, by providing "containers" [11].

Containers help the developers to concentrate on business logic by providing complex level functionality. Several libraries are added to J2EE specification to support Web Services [11].

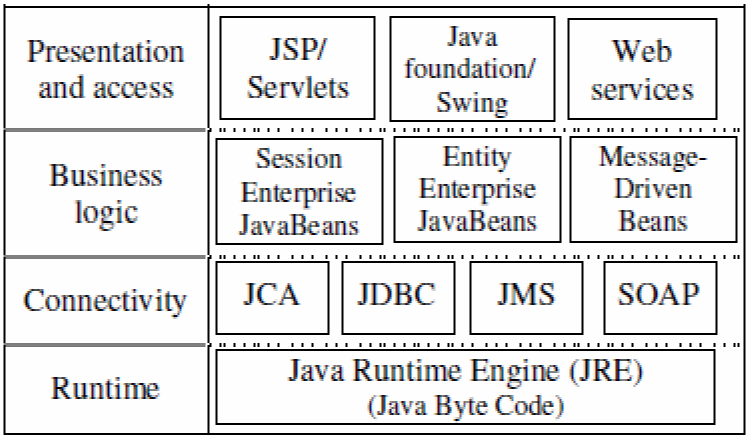


Figure 1: J2EE architecture [11]

- Java API for XML based RPC (JAX-RPC) is the API that enables developers to develop and deploy Web Services [11].
- Java API for XML registeries, (JAXR) provides a uniform and standard API to access different kind of XML registeries [11].

J2EE is the leader in the development of enterprise applications. The attractive choices of this platform are as follows.

Platform Independence: Java technology works independently of any platform or an operating system. The development platform is designed for Windows, Mac, and Solaris and to the flavors of UNIX like HP-UNIX [11].

Multi- vendor Support: J2EE compatibility test suite is provided by Sun Microsystems which ensures for compatibility among applications vendors which help to ensure portability for applications and components written in J2EE [11].

2.4.2 In .NET:

.NET is a Microsoft product that that is tied closely to the windows operating system. Microsoft describes it as the software that connects information, people, systems and devices.

Figure 2 shows the .NET development platform which is similar to J2EE for multi tier applications.

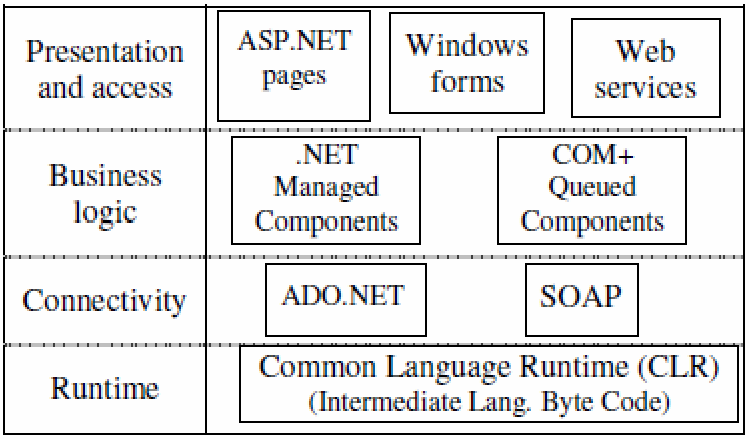


Figure 2: ASP.NET Web Service architecture [11]

Being the successor of the oldest Microsoft technologies such as DCOM and COM, several new features have been added to support Microsoft components.

Language independence: In terms of development language there is a vast choice to choose among programming languages like C#, VB.NET, Jscript.NET, C++ all which are part of .NET [11].

Integrated Web Services Support: The built in support for developing and deploying Web Services, make them appear just like any other objects. There is no visible difference for the programmer after creating a reference. Hence, the developing, publishing and discovery of the Web Services made simple, is one of the greatest advantage with .NET.

Previously Web Services with ASP.NET were language independent but could be used with windows machines only. However, with the recent development of Windows Communication foundation, the barrier is broken for cross platform communication. Applications built on other technologies, such as Java EE application servers, and applications running on Windows machines or on machines running other operating systems, such as Sun Solaris, IBM z/OS, or Linux can be communicated [11]

The KMIT is already developed in the .NET framework - and VB.NET language. So it's a good idea to continue with the tool that gets along with the existing framework. As described above, the KMIT Hotline Service which is developed in C# encounters no problem with the existing infrastructure considering the beauty of .NET to support multiple languages. With WCF added to the KMIT Hotline Service it also solves the problem for interoperability. More details about the integration of WCF to ASP.NET are described in the later sections.

3.0 WINDOWS COMMUNICATION FOUNDATION

3.1 About WCF

WCF is the unified programming model used to build service oriented applications. Web Services which are now universally accepted as a change to software development have standard protocols for application to application communication. Few of the services provided by Web Services apart from simple application integration include security, reliability, and transaction co-ordination. The benefits of the changes provided by the Web Services should be reflected in the tools and technologies used to develop them. Windows Communication Foundation (WCF) is designed to offer a manageable approach to distributed computing, broad interoperability, and direct support for service orientation [12].

WCF has well designed service model which enables the programmers to not only develop expertise ASP.NET applications but also helps the developers having familiar experience with .NET remoting, and enterprise services. The service model features a straightforward mapping of Web Services concepts to those of the .NET Framework common language runtime (CLR), including flexible and extensible mapping of messages to service implementations in languages such as Visual C# or Visual Basic [12]. Serialization facilities developed in WCF enable loose coupling and versioning, and provide integration with existing .NET distributed technologies such as message Queuing (MSMQ), ASP.NET Web Services and Web Service Enhancements (WSE). WCF is implemented primarily as a set of classes on top of the

.NET Framework CLR. Because it extends their familiar environment, WCF enables developers who create object-oriented applications using the .NET Framework today to also build service-oriented applications in a familiar way.

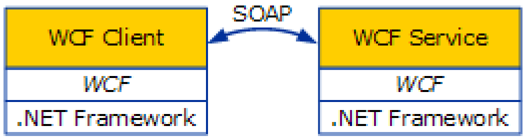


Figure 3: View of WCF client and service [12]

The above figure shows the client and the service communicating using SOAP, the native WCF message representation. Even though it shows that both of the applications are built using WCF, it is not required as compulsory.

3.2 Features of WCF

Many features have been added into WCF which makes the job of the developer of the services easier. Serialization and Hosting of services are noteworthy and these features are used in the development of Hotline Web Services. Detailed description is summarized below.

3.2.1 Serialization:

WCF has been built up around the tenets of service orientation. It supports several serialization mechanisms that make it easy to bring existing types forward and provides a simple, interoperable foundation for future service-oriented applications. While we can build XML directly, there is also an advantage to leverage the concepts of serialization mechanisms that can automate moving of objects between .NET framework and XML sets [13].

Generally there are two procedures that can be used for implementation of web services. One is to embrace XML and program directly to the messages, which offers high degree of flexibility, and the second is to predefine the mapping between XML and .NET and then rely on automated serialization mechanisms. This simplifies the developer experience by hiding various XML details. Both of them are supported with equal depth in WCF [13].

WCF represents all the messages using the message class found in 'System. ServiceModel. Channels'. The message class models a SOAP message. The SOAP message is commonly packaged as SOAP envelope, which has a header and a body section, using either 'System.Xml' classes or type-based serialization [13].

We can explicitly choose the method we want to use at message level. Using service contracts in terms of serializable types is the most common way of using serialization in WCF. An Example: 1 [13] can be shown below.

```
[ServiceContract]
public interface IEchoService
{
    [OperationContract]
    Person EchoPerson(Person person);
}
```

Example: 1

The .NET type definition will serve as service contract, by adding [servicecontract] to the .NET interface. Annotating the method signature with [OperationContract] indicates that the method is included in the service contract. At run time, windows communication foundation automatically maps the method signature to a pair of messages behind the scenes, each containing a “Person” in the SOAP body. It then uses a serializer to map the “Person” object into the message. (For complete control over what goes where in the SOAP envelope, [MessageContract] is used in types in the signature [13].

3.2.2 Hosting:

When the development relies on SOA architecture, then the service has to be robust. WCF offers a variety of hosting options which can help services robustness. The availability requirements of the service, managing and deployment of services, support to the older versions of services are to be taken into account before hosting. Windows communication foundation doesn't come with its own host, but instead comes with a class called ‘**ServiceHost’** that allows hosting WCF services in its own application. The application need not have to consider any of the network transport specifics to be able to make sure that your services are reachable. It's a matter of configuring the services' endpoints either programmatically or declaratively, and calling the ‘**Open’** method of ‘**ServiceHost’**. All of the generic functionality regarding bindings, channels, dispatchers, and listeners are integrated into ‘**ServiceHostBase’** and ‘**ServiceHost’**. This means that the responsibility of the application that you use to host your service, the application where ‘**ServiceHost’** is running, is significantly less than you would expect up front [14].

A WCF application requires a hosting windows process. Multiple .NET applications can be hosted in a single windows process. An application domain is the means for the .NET CLR to isolate the managed code from Windows. The CLR automatically creates one default application domain in each worker process where it is initialized in a process. The default application domain is not unloaded until the process in which it runs shuts down. The CLR controls the shutdown of the default application domain. In most hosts, no code is running inside the default application domain. Instead, hosts create a new application domain so the application domain can be closed independently of the process. In a lot of applications, it is desirable that the client-side code and server-side code execute in different application domains. Often, these desires stem from reasons such as security and isolation.

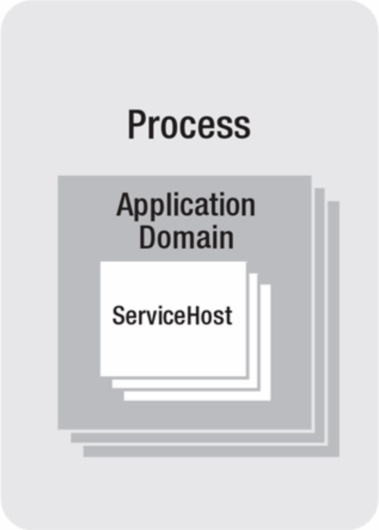


Figure 4: Application domain of process [14]

Figure 4 shows that every process has at least one application domain, and each application domain can host zero or more WCF '**ServiceHost'** instances. WCF requires at least an application domain hosted inside a Windows process [14]

The various techniques available in WCF to host a service are

- **Self Hosting a Service**: self hosting is the easiest and flexible way to host a service. Two requirements are to be met to self host a service. WCF

- **Hosting in Windows Services**: A windows service is the process managed by the operating system. Windows comes with the service control manager, which controls the services installed on the operating system. Windows uses services to support operating system features such as networking, USB, remote access, message queuing, and so on. It is to be noted that windows service is not same as web service [14].
- **Hosting using IIS**: Hosting a WCF Service in IIS, needs a new physical file with the .svc extension. The file associates a service with its implementation and is the means for IIS to create 'ServiceHost'. IIS takes over the interaction between the service and 'ServiceHost'. No initiation is required to start the 'ServiceHost'. The first line of the .svc file contains a directive enclosed in the ASP.NET **'<% Page %>'** directive that tells the hosting environment to which service this file points. The most common scenario is to define endpoints in a configuration file. In IIS, the endpoints are defined in the 'Web.config' file. In IIS, web configuration files can be nested in sites, applications, and virtual directories. WCF takes all the configuration files into account and merges services and their endpoints together. This means that nested 'Web.config' files are additive to each other, where the last file read in the bottom of the hierarchy takes precedence over files higher in the hierarchy [14].

3.3 Security in WCF:

WCF is a distributed programming platform based on SOAP messages. Using WCF, you can create applications that function as both services and service clients, creating and processing messages from an unlimited number of other services and clients. In such a distributed application, messages can flow from node to node, through firewalls, onto the Internet, and through numerous SOAP intermediaries. This introduces a variety of message security threats. WCF uses concepts that are familiar to build secure, distributed applications with existing technologies such as HTTPS, Windows integrated security, or user names and passwords to authenticate users. WCF not only integrates with existing security infrastructures, but also extends distributed security beyond 'Windows only' domains by using secure SOAP messages. Consider WCF an implementation of existing security mechanisms with the major advantage of using SOAP as the protocol in addition to existing protocols. For example, credentials that identifies a client or a service, such as user name and password or X.509 certificates, have interoperable XML-based SOAP profiles. Using these profiles, messages are exchanged securely by taking advantage of open specifications like XML digital signatures and XML encryption.

3.3.1 Common security threats:

The following examples illustrate some common threats that WCF security can help mitigate when exchanging messages between entities**:**

- Observation of network traffic to obtain sensitive information. For example, in an online-banking scenario, a client requests the transfer of funds

- Rogue entities acting as services without awareness of the client. In this scenario, a malicious user (the rogue) acts as an online service and intercepts messages from the client to obtain sensitive information. Then the rogue uses the stolen data to transfer funds from the compromised account. This attack is also known a *phishing attack* [15].
- Alteration of messages to obtain a different result than the caller intended. For example, altering the account number to which a deposit is made allows the funds to go to a rogue account [15].
- Hacker replays in which a nuisance hacker replays the same purchase order. For example, an online bookstore receives hundreds of orders and sends the books to a customer who has not ordered those [15].
- Inability of a service to authenticate a client. In this case, the service cannot assure that the appropriate person performed the transaction [15].

3.3.2 Integration of WCF with existing authentication models

Most important aspect of a security model is to provide proper authentication between the entities of communication. Digital signatures or credentials are the common forms of authentication among the communicating peers. But with the improvement in distributed computing scenarios, various authentication mechanisms have emerged. Thus, in the world of Web Services, where the same service might be exposed to internal customers as well as to external partners or Internet customers, it is important that the infrastructure provide for integration with these existing security authentication models. In this section we describe the various authentication methods that can be used with WCF.

- **Anonymous caller**: When using this option, the WCF service does not authenticate the callers. This may not be the recommended option from a security perspective [15].
- **Username with client credential**: When using this option, the caller provides a username and password to the service. The service can authenticate against windows credentials, or a membership provider such as the 'Microsoft SQL Server membership provider', or use a custom validator to validate against the custom store. This option is recommended only when windows authentication is not possible. The service is authenticated by using a service certificate [15].
- **Certificate client credential**: When using this option, the caller presents an X.509 client certificate. The WCF service looks up the certificate information on the host side and validates it (peer trust), or trusts the issuer of the client certificate (chain trust). This option should be used when windows authentication is not possible, or in the case of B2B scenarios. The service is authenticated by using a service certificate [15].
- **Windows**. When using this option, the WCF service uses Kerberos authentication when in a domain, or NTLM authentication when deployed in a workgroup environment. This option uses the windows token

For the KMIT Hotline Service, each client application is provided with a user name and password pair which are stored and validated against the database using SQL server membership provider. Hence, the second authentication model described above is used for developing Hotline Web Service, where the service uses an X.509 certificate to authenticate itself. The implementation is described in detail in later chapters.

3.3.3 Standards and Interoperability:

Distributed computing/communications platforms need to interoperate with the technologies different vendors offer. With large and different deployments, maintain homogeneity and interoperable security could be an issue. In order to maintain interoperability security among distributed systems, companies active in Web Service industry, have made some standards. Regarding Security, few notable standards like WS-Security: SOAP Message Security (accepted by the OASIS standards body and formerly known as WS-Security), [18] WS-Trust [21], WS-SecureConversation [22], and WS-SecurityPolicy are proposed [23].

WCF supports a wide variety of interoperability scenarios. The 'BasicHttpBinding' class is targeted at the basic security profile and the 'WSHttpBinding' class is targeted at the latest security standards, such as WS-Security 1.1 and WS-SecureConversation. By adhering to these standards, WCF security can interoperate and integrate with Web Services that are hosted on operating systems and platforms other than Microsoft windows.

3.3.4 WCF security functional areas:

WCF security functional areas are divided into three main aspects relating to transfer security, access control and auditing. This section describes in detail about the transfer security, that which is more relevant and implemented in the development of Hotline Web Service. The whole of transfer security talks about three main important aspects of message transfer namely, authentication, integrity and confidentiality. Authentication is the ability to verify a claimed identity. Integrity checks if the message is being tampered or not, and confidentiality is the ability to allow only the recipient to read the message that's transferred. The two main modes in which the transfer security is implemented is transport and message security.

- **Transport security**: Transport layer security represents an approach where the underlying operating system or application servers are used to handle security features. For data confidentiality, Secure Sockets Layer (SSL) is a common transport layer approach that is used to provide encryption. Figure 4 below shows the security [24].

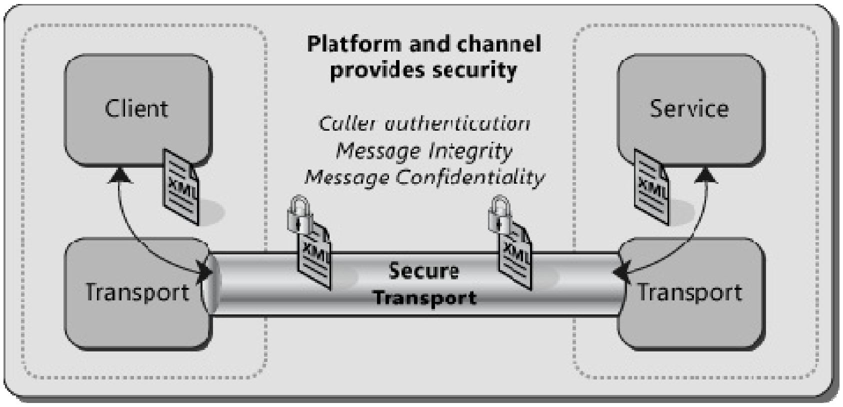


Figure 5: Transport layer security [25]

TLS uses a transport-level protocol, such as HTTPS, to achieve transfer security. Transport mode has the advantage of being widely adopted, available on many platforms, and less computationally complex. If a message needs to go through multiple points to reach its destination, each intermediate point must forward the message over a new SSL connection. In this model, the original message from the client is not cryptographically protected on each intermediary because it traverses intermediate servers and additional computationally expensive cryptographic operations are performed for every new SSL connection that is established [24]. This is the disadvantage of TLS i.e. securing messages only from point-to-point.

- **Message layer security:** Message layer security represents an approach where all the information related to security is encapsulated in the SOAP message. There are many advantages of using message layer security to transport layer security. Some of them are

   1. **Increased flexibility**. Parts of the message, instead of the entire message, can be signed or encrypted. This means that intermediaries can view the parts of the message that are intended for them. An

2. **Support for auditing**. Intermediaries can add their own headers to the message and sign them for the purpose of audit logging [24].

3. **Support for multiple protocols**. Applications can send secured messages over many different protocols such as Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP), and Transmission Control Protocol (TCP) without having to rely on the protocol for security [24].

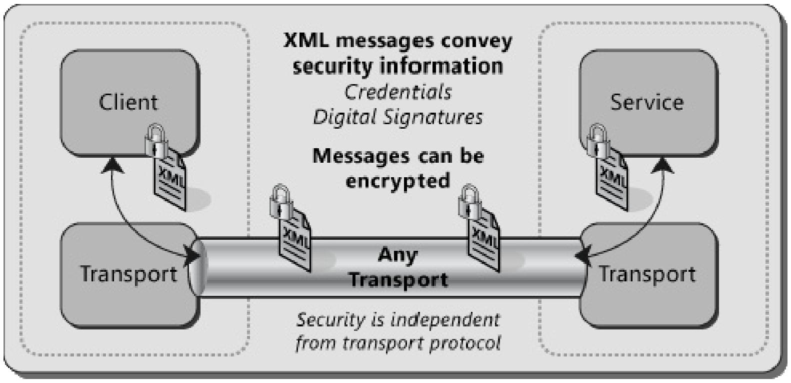

XML messages convey security information
Credentials
Digital Signatures
Messages can be encrypted
Client
Service
XML
Transport
Any Transport
Transport
Security is independent from transport protocol

Figure 6: Figure for message security [25]

Even though message level security has many advantages to transport layer security, it has to be considered only when message is being routed through intermediaries before it reaches the destination. This is because the performance of message layer security is less when compared to TLS. The specifications and future needs for the project are to be considered while choosing the specific mode of security implementation. KMIT Hotline Web Service displays information from Hotline Service to its client applications. Even though the route does not consist of intermediaries, the Web Service is designed keeping in view the future needs of the clients where there are more chances to route the messages through intermediaries. The method of implementation of message security to KMIT Hotline is discussed in detail in further chapters.

4.0 MOTIVATION

Evolutions are a way of life in the computer industry. Only 20 years ago, the world was still in the mainframe era. Few people had access or used computers, and when they did, it was only through the nearest computer center. Three innovations changed all that: the PC, the GUI, and the Internet. Since then, standards such as HTML and HTTP have exponentially increased people's use of the Internet. This base protocol for viewing content on the web grew web usage to what we are familiar with today. The Web became a key activity in the daily lives of businesses, employees, and consumers. Many of us envision an online world where constellations of PCs, servers, smart devices, and Internet-based services can collaborate seamlessly. Businesses will be able to share data, integrate their processes, and join forces to offer customized, comprehensive solutions to their customers. The information you or your business need will be available wherever you are, and whatever computing device, platform, or application you are using.

Today's standalone applications and Web sites create islands of functionality and data, which has to be navigated manually between web sites, devices, and applications, logging in each time, and rarely being able to carry data with! Tasks that ought to be simple, such as arranging a meeting with colleagues from partner companies and automatically updating every attendee's calendar, are a nightmare in the best case, and impossible in the common case. This inefficiency is a major source for productivity loss. As a result of the changes in how businesses and consumers use the web, the industry is converging on a new computing model that enables a standard way of building applications and processes to connect and exchange information over the web. This new Internet-based integration methodology, called "XML Web Services," enables applications, machines, and business processes to work together in a revolutionary way. The widespread support around XML assures that businesses will cooperate in the Internet-based economy with this XML Web Services model.

At the heart of the solution is XML (extensible Markup Language). XML is an open industry standard managed by the World Wide Web Consortium. It enables developers to describe data being exchanged between PCs, smart devices, applications, and web sites. Since the data is separate from the format and style definitions, it can be easily organized, programmed, edited, and exchanged between any web sites, applications, and devices. Just as the web revolutionized how users talk to applications, XML transforms how applications talk to each other. *[Some arguments taken from Bill Gates, in his leaflet to Developers & IT Professionals from June 14, 2001, covers the motivation for Web services, and the XML's role in this technology]*

KMIT is a knowledge management information tool that is custom built for the D&D community. This system is being developed by Florida International University- Applied Research Center in collaboration with Department of Energy (DOE EM 20), EFCOG and ALARA centers at Hanford and Savannah River. The Hotline Module of KMIT allows interested users to post questions/problems related to specific area of interest in the area of Decontamination and Decommissioning (D&D). The question/problem will be routed to a preselected subject matter specialist

(SMS) who, based on his/her experience, will provide a technical solution to the posted question/problem. The provided answer will be posted on a web portal after content coordinator (CC) review. Various clients working with DOE and KMIT want to display the latest published problems of KMIT Hotline search in their own applications on a regular basis. Considering one of the major benefits of Web Services is the ease of integration of one piece of software with another, we proposed the idea of developing Web Service for KMIT Hotline module which can be plugged into other client's applications. This ease of integration will enable tighter business relationships and more efficient business processes. An integral part of the XML Web Services programming model, is the ease of integration with external data sources. No longer does KMIT client applications need to copy and maintain external data sources. Applications can request and get information in real time, and transform into a particular format. This will allow the client applications to deliver individualized software and services, while the maintenance burden is reduced. Hotline Web Service will be an integrated experience that excels in its simplicity. It gives users the ability to act on information any time, any place, and from any smart device. Businesses will love KMIT Web Service because it will force them to streamline their processes. An XML Web Service is a simple, reliable way to blend existing systems with new applications and services.

5.0 WS- SECURITY PROTOCOL:

5.1 WS-Security model:

This security protocol describes enhancements to SOAP messaging to provide message authentication integrity and confidentiality. The specified mechanisms can be used to accommodate a wide variety of security models and encryption technologies. This protocol also provides a general-purpose mechanism for associating security tokens with message content. No specific type of security token is required, the specification is designed to be extensible (i.e. support multiple security token formats).

This OASIS specification is the result of significant new work by the WSS Technical Committee and supersedes the input submissions, Web Service Security (WS-Security) Version 1.0 [16] and Web Services Security Addendum Version 1.0 [17] This protocol specification proposes a standard set of SOAP extensions that can be used when building secure Web Services to implement message content integrity and confidentiality. This specification refers to this set of extensions and modules as the "Web Services Security: SOAP Message Security" or "WSS: SOAP Message Security". This specification is flexible and is designed to be used as the basis for securing Web Services within a wide variety of security models including PKI, Kerberos, and SSL. Specifically, this specification provides support for multiple security token formats, multiple trust domains, multiple signature formats, and multiple encryption technologies. The token formats and semantics for using these are defined in the associated profile documents. This specification provides three main mechanisms: ability to send security tokens as part of a message, message integrity, and message confidentiality. These mechanisms by themselves do not provide a complete security solution for Web Services. Instead, this specification is a building block that can be used in conjunction with other Web Service extensions and higher-level application-specific protocols to accommodate a wide variety of security models and security technologies. These mechanisms can be used independently (e.g., to pass a security token) or in a tightly coupled manner (e.g., signing and encrypting a message or part of a message and providing a security token or token path associated with the keys used for signing and encryption). [18]

The foundation of the Message security mode is the *WS-Security* specification. The WS-Security specification defines a framework that allows security to be applied to SOAP messages. It specifies a message security model using security tokens combined with digital signatures and encryption to protect and authenticate SOAP messages. Some important terminology is mentioned below,

- A *security token* asserts claims and can be used to assert the binding between authentication secrets or keys and security identities [19].
- A *claim* is a declaration made by an entity about an entity (for example, a name, identity, group, key, group, or privilege). The entity that makes the claim is referred to as a *claim issuer*; the entity about which the claim is made is referred to as a *claim subject* [19].

- A *claim issuer* can vouch for or endorse the claims in a security token by using its key to sign or encrypt the security token. This enables authentication of the claims in the security token [18].
- *Message signatures* are used to verify message origin and integrity. Message signatures are also used by message producers to demonstrate knowledge of the key, typically from a third party, used to confirm the claims in a security token and thus to bind their identity (and any other claims represented by the security token) to the messages they create [18]
- *Confidentiality* is the property that data is not made available to unauthorized individuals, entities, or processes [18].
- A *digital signature* is a value computed with a cryptographic algorithm and bound to data in such a way that intended recipients of the data can use the digital signature to verify that the data has not been altered and/or has originated from the signer of the message, providing message integrity and authentication. The digital signature can be computed and verified with symmetric key algorithms, where the same key is used for signing and verifying, or with asymmetric key algorithms, where different keys are used for signing and verifying (a private and public key pair are used) [18].
- *End-to-end message level security* is established when a message that traverses multiple applications (one or more SOAP intermediaries) within and between business entities, e.g. companies, divisions and business units, is secure over its full route through and between those business entities [18].

- *Trust is* the characteristic that one entity is willing to rely upon a second entity to execute a set of actions and/or to make set of assertions about a set of subjects and/or scopes [18].

WS-Security defines several types of security tokens and gives an extensible model that allows additional security token types to be defined independently. Every token type definition contains a XML serialization of the token. This allows adding the token representation directly to the message [19]

The following are some of the security token types defined in WS-Security:

- Username Token.
- X.509 Certificate Token.
- Kerberos Token.
- SAML Token.

In .NET Framework 3.0, a client message can contain only one token of any given type, but can contain tokens of different types. In .NET Framework 3.5, client messages can contain multiple tokens of a given type, as well as tokens of different types.

5.2 WS – Security implementation in WCF

Because WS-Security lays a foundation for message security, the WCF implementation of WS-Security is a cornerstone of the whole Message security mode. To extend the Message security mode functionality, it is necessary to understand how the WS-Security implementation works.

The WS-Security implementation in WCF handles the following:

- Serialization of security tokens to and from SOAP messages.
- Authentication of security tokens.
- Application and verification of message signatures.
- Encryption and decryption of SOAP messages.

WCF extensibility points allow customization of the first two items. It is possible to change the serialization of existing security tokens or the way WCF security authenticates those tokens. It is also possible to introduce completely new security token types to the WCF security, including the serialization and authentication functionality. The following topics in this section show how the WS-Security implementation extensibility points can be used to customize the security tokens functionality.

Authentication: Security tokens are desterilized from the incoming message and authenticated. The authentication process results in a set of authorization-policy objects. Each object represents a part of the security token's data. That data is used during the authorization stage [19].

Identity: WCF creates an implementation of the 'IIdentity' interface to represent the caller to the existing infrastructure (created by the .NET Framework security model). This 'IIdentity' instance represents either the Windows identity of the caller if the security token is mapped to a Windows account, or a Primary Identity that contains the caller name. Those identities are also accessible using the 'ServiceSecurityContext'. It is possible to customize the way identities are created in WCF. The custom membership provider works only if user name/password authentication is used to authenticate the caller. The 'MembershipProvider' validates the user name/password pair. If the pair is valid, WCF creates a Primary Identity that represents the authenticated caller after ‘MembershipProvider’ validation [19].

5.3 Sending secure messages:

The following steps describe how a message is secured on the client when using the message security mode. The illustration shows which components are involved and the relationships between them.

- The client application runs and generates a message.
- In the *Token Provisioning* stage, the client credentials (username/password in Hotline Service scenario) are attached.
- These credentials are used to create the security token.
- In the *Token Authentication* stage, the tokens are verified.
- Finally, the security tokens are serialized and sent [19].

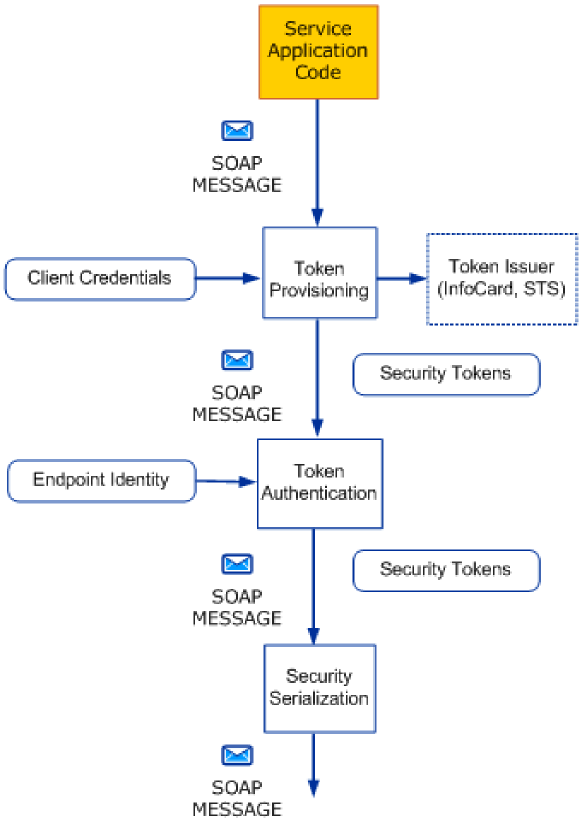


Figure 7: Steps to secure the messages on client [19]

5.4 Receiving secure messages:

The following illustration shows the processes that occur when a secure message is extracted from the wire and verified on the receiving side:

- The security tokens are deserialized and processed in the token authentication stage. ASP.NET membership provider can be used to supply user names and passwords generated from the client at this point [19].
- After authentication, the authorization policies are extracted. (optional) [19].
- In the Authorization Policies Evaluation stage, the authorization policies are evaluated and claims can be added to an Evaluation Context. External authorization policies are also used at this point. This step, as well as the

- In the Service Authorization stage, the correct authorizations are given based on claims added by the authorization policies. This step is done by methods of the 'ServiceAuthorizationManager' [19].
- WCF generates a 'PrincipalPermission' using the credentials at this point. If required, an ASP.NET role provider can be used at this point[19]
- The application code runs.

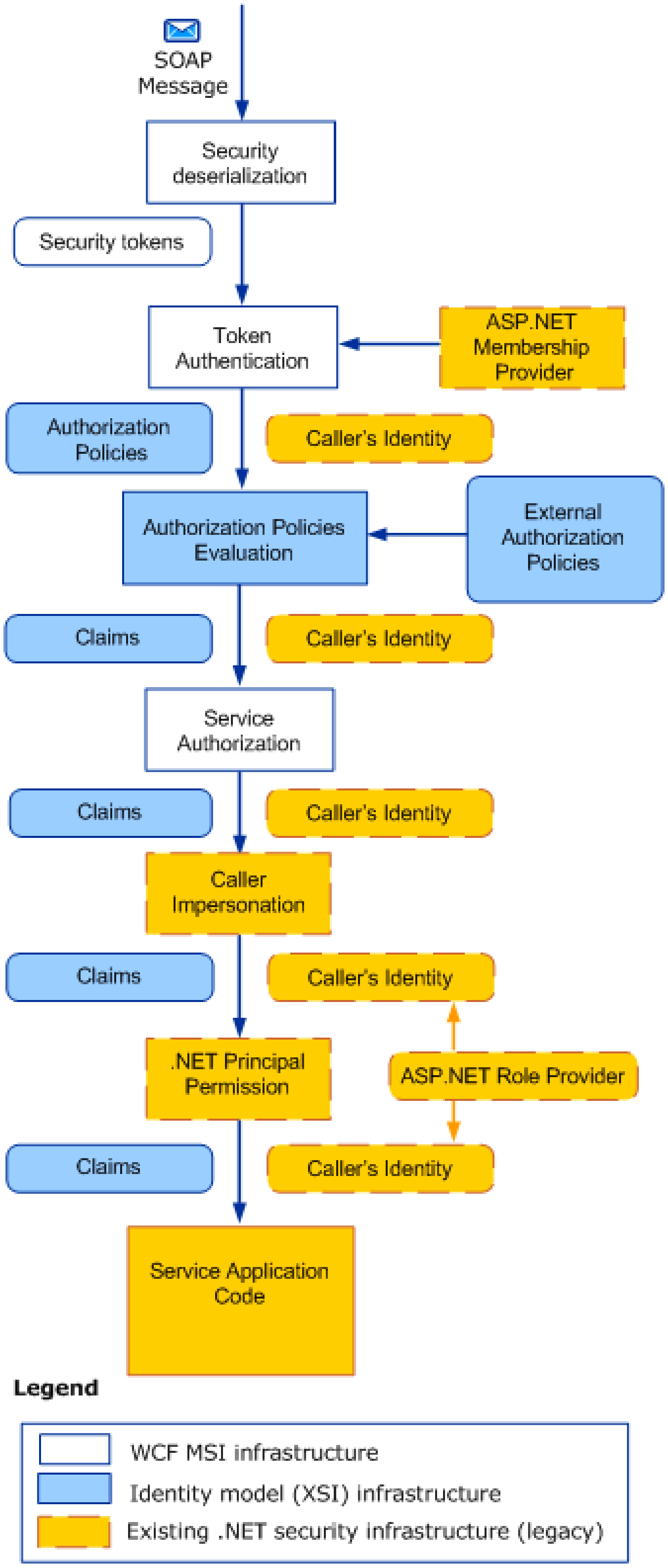


Figure 8: Receiving secure messages [19]

6.0 HOTLINE SERVICE FOR KMIT

This chapter delves into the implementation details of the HOTLINE WEBSERVICE for KMIT. Introduction about WCF basics is provided, which are later used while writing the service. The Hotline Service for KMIT is implemented in three steps. One is development of the service, which talks about what are service and data contracts, the endpoints and binding information. The second step is hosting of the service. This step picks one of the hosting mechanisms discussed in chapter 3.2.2 and explains the process of hosting the Hotline Service. The third step talks about how a client application can consume the service and integrate it into its own application that can be used for further communication. As discussed in 3.1 WCF is the .NET 3.0 way of communicating between objects across machines. It is based on WSDL. WCF service requires a couple of things to be done right before it starts working successfully. Let's look at the basics which are kept as straight as possible.

6.1 Basics of WCF:

The basics of WCF are popularly known as ABC's of WCF. Where 'A' stands for address, 'B' for binding, and 'C' for contract information of the service.

- *Address* is *where* you communicate. This is not the same as the location to deploy the service, but the URL that will be used internally to map your requests and responses.
- *Binding* is *how* to communicate with service. There are several default ones like 'BasicHttp', 'TCP', 'NamedPipes', 'MSMQ' and several others. This is the protocol that the server and client understands while communicating.

- *Contract* is *what* to communicate. It defines the functionality provided by the service to its clients. This contract information is of two types. Service contract and Data contract.
- *Service Contract* is the API the service consumer invokes on the service. It's the method signature that will go into the WSDL. *Data Contracts* is the data that would travel from the service consumer and the service. It's the data structure. This can be found in the schema of the service.

Now is the moment to define what basically a service is. A service is a construct that exposes one or more *endpoints,* each of which exposes one or more service operations. An endpoint is the operation that gets exposed which contain the ABC's defined.

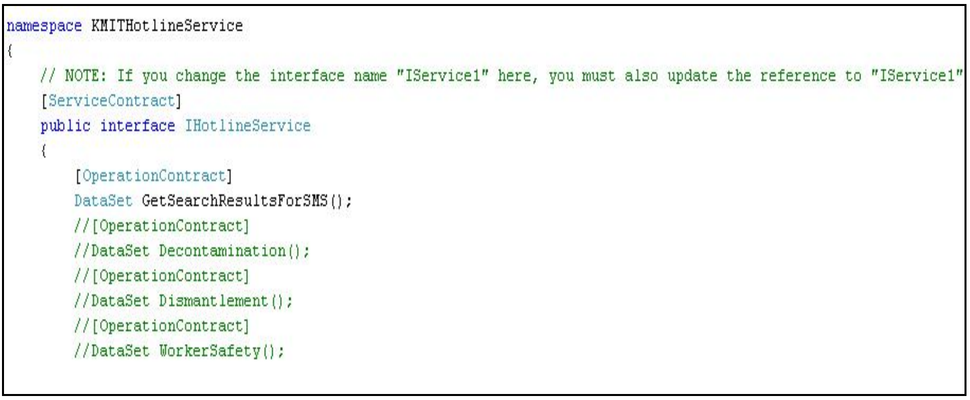
6.2 Development of the Hotline Service:

6.2.1 Designing a service contract for the Hotline Service

Services are groups of operations. To create a Hotline Service contract it must model operations and specify their grouping. In Windows Communication Foundation (WCF) applications, define the operations by creating a method and marking it with the 'OperationContractAttribute' attribute. Then, create a service contract; group together the operations, either by declaring them within an interface marked with the 'ServiceContractAttribute' attribute, or by defining them in a class marked with the same attribute.

The following steps define creating a service:

- In Visual Studio on the **File** menu, click **New Project**.
- In the **Templates** section, select **WCF Service**. Make sure that the **Location** is set and specify the virtual directory to be created in the **Path** (C:\Documents and settings\harini\My Documents\Visual Studio 2008\Projects\KMITHotlineService\KMITHotlineService\)
  - The interface is shown in figure



```
namespace KMITHotlineService
{
    // NOTE: If you change the interface name "IService1" here, you must also update the reference to "IService1"
    [ServiceContract]
    public interface IHotlineService
    {
        [OperationContract]
        DataSet GetSearchResultsForSMS();
        //[OperationContract]
        //DataSet Decontamination();
        //[OperationContract]
        //DataSet Dismantlement();
        //[OperationContract]
        //DataSet WorkerSafety();
```

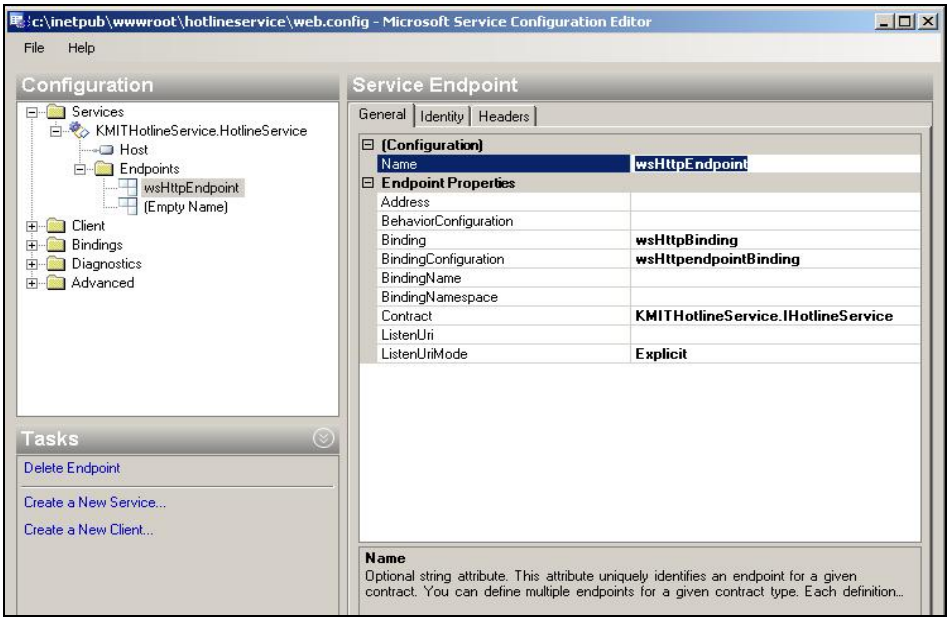
9.

Figure 9: Interface for KMIT Hotline Service

The service contract exposes the interface of the Hotline Service and operation contract describes the operations the service exposes. Here, the Hotline Service exposes one operation ‘GetSearchResultsForSMS()’ which returns the top 5 problems of the Hotline module. The return value of the operation is a Dataset. ‘IHotlineService’ is the interface which exposes the ABC information required for the client application. WCF environment provides with a ‘web.config’ file that is automatically created when a service is defined. Changes made to the ‘web.config’ file reflect changes to the service. The changes can be done by manually adding the code to the file or by editing the information through ‘web.config’ interface. This reduces the burden of writing the code. The following describe the series of steps to configure ABC’s for Hotline service endpoint.

In the Solution Explorer, right-click the ‘Web.config’ of the WCF service, and choose the **Edit WCF Configuration** option. If you do not see the **Edit WCF Configuration** option, click the **Tools** menu and select **WCF Service Configuration Editor**. Close the WCF Service Configuration Editor tool that appears. The option should now appear on the ‘web.config’ context menu. In the Configuration Editor, in the **Configuration** section, expand **Service** and then expand **Endpoints**.

- Select the first node **[Empty Name]**. Set the **name** attribute to **wsHttpEndpoint**. By default, the name field will be empty because it is an optional attribute.

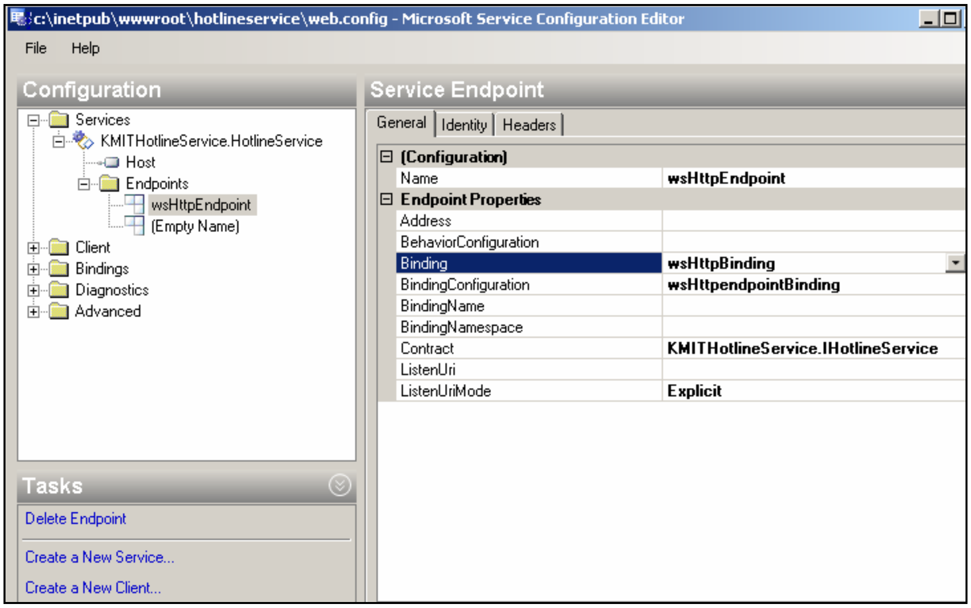


(Configuration)	
Name	wsHttpEndpoint
Endpoint Properties	
Address	
BehaviorConfiguration	
Binding	**wsHttpBinding**
BindingConfiguration	**wsHttpendpointBinding**
BindingName	
BindingNamespace	
Contract	**KMITHotlineService.IHotlineService**
ListenUri	
ListenUriMode	**Explicit**

Name
Optional string attribute. This attribute uniquely identifies an endpoint for a given contract. You can define multiple endpoints for a given contract type. Each definition...

Figure 10: Setting the name attribute to WSHttpEndpoint

- Click the **Identity** tab and then delete the **Dns** attribute value.
- In the Configuration Editor, select the **Bindings** folder.
- In the **Bindings** section, choose **New Binding Configuration**.
- In the **Create a New Binding** dialog box, select **wsHttpBinding**.
- Click **OK**.

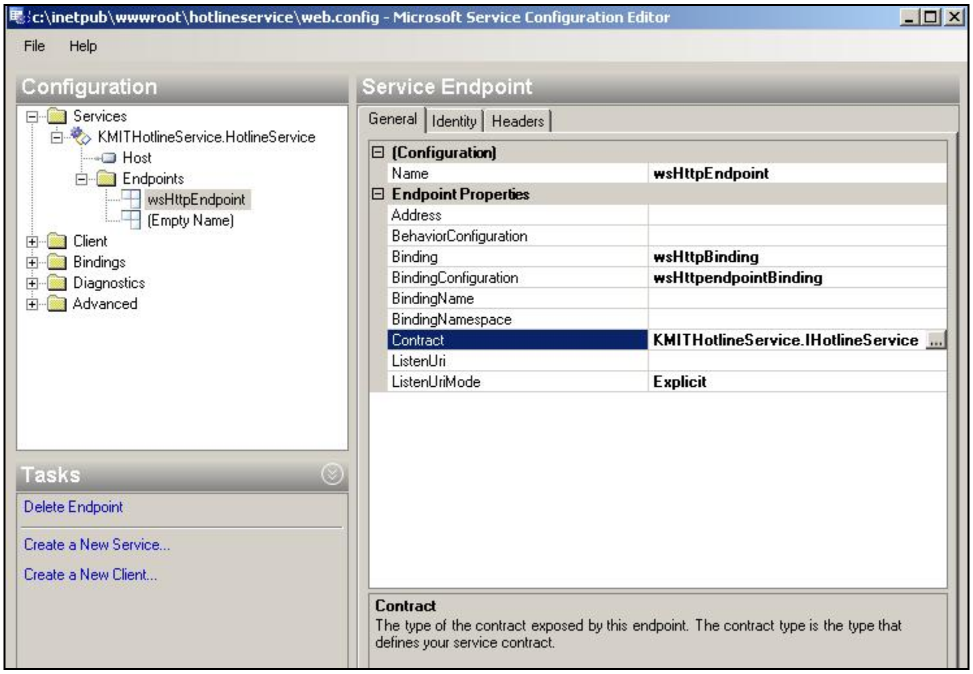


(Configuration)	
Name	wsHttpEndpoint
Endpoint Properties	
Address	
BehaviorConfiguration	
Binding	wsHttpBinding
BindingConfiguration	wsHttpendpointBinding
BindingName	
BindingNamespace	
Contract	KMITHotlineService.IHotlineService
ListenUri	
ListenUriMode	Explicit

Figure 11: Setting the binding configuration

- Set the **Name** of the binding configuration to some logical and recognizable name; for example, **wsHttpEndpointBinding**.
- Click the **Security** tab.
- Make sure that the **Mode** attribute is set to **Message**, which is the default setting. This is one of the parts of adding security to the service. Total security for Hotline Service is discussed in detail in later chapters)

- Set the **MessageClientCredentialType** to the **Username** option by selecting this option from the drop-down list. (As taken from chapter 3 to use username authentication for Hotline Service)
- In the **Configuration** section, select the **wsHttpEndpoint** node.
- Set the **BindingConfiguration** attribute to **wsHttpEndpointBinding** by selecting this option from the drop-down list.
- Set the **Contract** information to KMIT 'HotlineService.IHotlineService'



c:\inetpub\wwwroot\hotlineservice\web.config - Microsoft Service Configuration Editor

(Configuration)	
Name	**wsHttpEndpoint**
Endpoint Properties	
Address	
BehaviorConfiguration	
Binding	**wsHttpBinding**
BindingConfiguration	**wsHttpendpointBinding**
BindingName	
BindingNamespace	
Contract	**KMITHotlineService.IHotlineService**
ListenUri	
ListenUriMode	**Explicit**

Contract
The type of the contract exposed by this endpoint. The contract type is the type that defines your service contract.

Figure 12: Setting the contract information to KMIT Hotline Service

The 'web.config' file would look like this

```
</binding>
   </wsHttpBinding>
  </bindings>
            <services>
   <service behaviorConfiguration="ServiceBehavior"
name="KMITHotlineService.HotlineService">
    <endpoint address="" binding="wsHttpBinding"
bindingConfiguration="wsHttpendpointBinding"
     name="wsHttpEndpoint"
contract="KMITHotlineService.IHotlineService">
     <identity>

      <dns value="localhost" />


<endpoint address="mex" binding="mexHttpBinding"

contract="IMetadataExchange" />


   </service>
  </services>
```

6.2.2 Creating user for SQL Server Membership Provider

The SQL Server membership provider stores user information in a SQL Server database. You can create your SQL Server user store manually by using Aspnet_regsql.exe from the command line.From a Microsoft Visual Studio® 2008 command prompt, run the following command:

- Aspnet_regsql –S .\SQLExpress –E –A –m

In this command:

- **-S** specifies the server, which is (.\SQLExpress) in this example.
- **-E** specifies to use Windows authentication to connect to SQL Server.
- **-A m** specifies to add only the membership feature. For simple authentication against a SQL Server user store, only the membership feature is required.

6.2.3 Grant Access Permissions to WCF Process identity

The Hotline WCF service process identity requires access to the ‘aspnetdb’ database. If you host the WCF service in Internet Information Services (IIS) 6.0 on Microsoft Windows Server 2003, the NT AUTHORITY\Network Service account is used by default to run the WCF service.

To grant database access

- Create a SQL Server login for NT AUTHORITY\Network Service.
- Grant the login access to the ‘aspnetdb’ database by creating a database user.
- Add the user to the **aspnet_Membership_FullAccess** database role.

Steps are performed by using the SQL Server Enterprise Manager, or the following script is run in SQL Query Analyzer

```
- Create a SQL Server login for the Network Service account
sp_grantlogin 'ASPNET'

-- Grant the login access to the membership database
USE aspnetdb
GO
sp_grantdbaccess 'ASPNET'

-- Add user to database role
USE aspnetdb
GO
sp_addrolemember 'aspnet_Membership_FullAccess',
```

But, the Hotline service is running on Microsoft Windows XP, create a SQL Server login for the ASPNET identity instead of the NT Authority\Network Service identity, as the IIS process runs under the ASPNET account in Windows XP.

6.2.4 Configure Membership Provider for Username Authentication

In this step, you configure the SQL Server membership provider to use username authentication.

- In the ‘web.config’ file, replace the existing single **<connectionStrings/>** element with the following to point to your membership database:

```
<connectionStrings>
            <add name="KMITConn"
connectionString="Server=.\sqlexpress; Database=DOEKM; User
Id=doekmadmin; password= doekmadmin@Miami;"
providerName="System.Data.SqlClient"/>
        </connectionStrings>
```

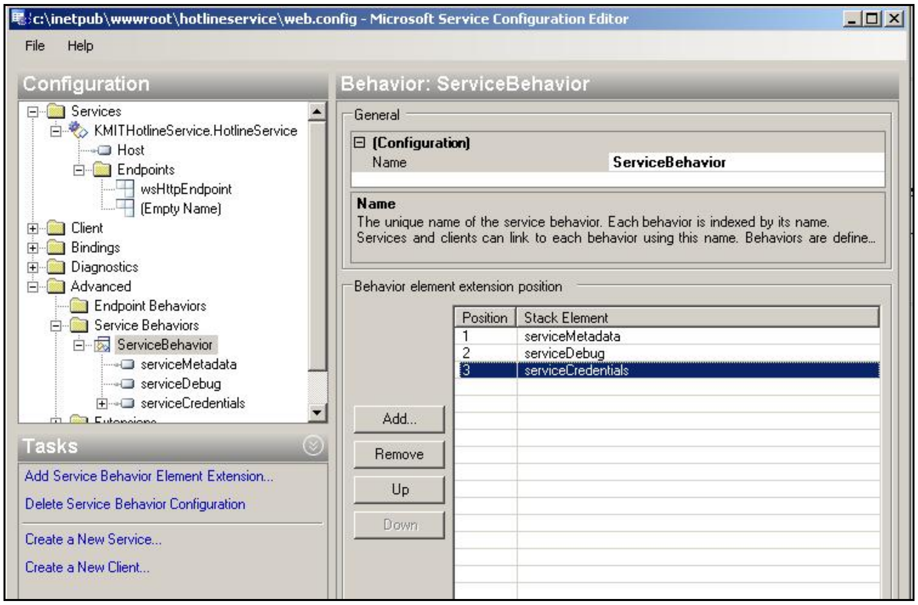
Where name ‘KMITConn’ represents the name of the connection used to connect to the database, ‘ConnectionString’ describes that the server being used is ‘sqlexpress’ and the database used is ‘DOEKM’ , which has ‘SQLserver’ authentication containing username and password credentials.

- Add a **<membership>** element inside the **<system.web>** element as shown in the following example. Note that the use of the **<clear/>** element prevents the default provider from being loaded and never used.

```
<membership defaultProvider="MySqlMembershipProvider">
<providers>
<clear/>
<add name="MySqlMembershipProvider"
connectionStringName="KMITConn"
applicationName="KMITHotlineService"
type="System.Web.Security.SqlMembershipProvider"/>
</providers>
</membership>
```

The code snippet mentions that the default provider used for KMIT Hotline Service is SQL ‘MembershipProvider’ and the connection string is ‘KMITConn’ as mentioned in the previous section. This specifies that the membership provider is being used for KMITHotlineService.

- Save the ‘web.config’ file, to ensure that the changes do not get lost during the following steps.
- In the configuration editor, expand the **Advanced** node, and then expand the Service Behaviors folder.
- Select the default behavior that was created with name ServiceBehavior.
- In the **Behavior: ServiceBehavior** section, click **Add**.
- In the **Adding Behavior Element Extension Sections** dialog box, select **serviceCredentials** and then click **Add**.
- In the **Configuration** section, under **Service Behaviors**, select the **serviceCredentials** option.



(Configuration)

Name ServiceBehavior

Name
The unique name of the service behavior. Each behavior is indexed by its name. Services and clients can link to each behavior using this name. Behaviors are define...

Position	Stack Element
1	serviceMetadata
2	serviceDebug
3	serviceCredentials

Figure 13: WCF configuration editor selects ServiceCredentials

- Set the **UsernamePasswordValidationMode** attribute to **MembershipProvider** by choosing this option from the drop-down list.
- Set the **MembershipProviderName** attribute to **MySqlMembershipProvider**.
- In the configuration editor dialog box, on the **File** menu, select **Save**.
- In visual studio, verify your configuration. The configuration should look as follows

```
<behaviors
<serviceBehaviors>
<behavior name="ServiceBehavior">
<serviceMetadata httpGetEnabled="true"/>
<serviceDebug includeExceptionDetailInFaults="false"/>
<userNameAuthentication
serNamePasswordValidationMode="MembershipProvider"
membershipProviderName="MySqlMembershipProvider"/>
</serviceCredentials>
</behavior>
```

These steps describe the development of Hotline Service and adding a user to the SQL database and configuring the membership provoder to validate the client credentials against the database, using username and pawwsord authentication.

7.0 SECURITY MODEL FOR HOTLINE SERVICE

The Hotline Service developed should also be configured with message security, so that the client credentials that are being transferred to the service are authenticated against the database. Adding message security will ensure that the credentials are encrypted and signed before they are being transmitted over as a SOAP message. Security tokens assert claims and signatures provide a mechanism for proving the sender's knowledge of the key. As well, the signature can be used to 'bind' or 'associate' the signature with the claims in the security token (assuming the token is trusted). Note that such a binding is limited to those elements covered by the signature.

A claim can be either endorsed or unendorsed by a trusted authority. A set of endorsed claims is usually represented as a signed security token that is digitally signed or encrypted by the authority. An X.509 certificate, claiming the binding between one's identity and public key, is an example of a signed security token. An endorsed claim can also be represented as a reference to an authority so that the receiver can 'pull' the claim from the referenced authority.

For this a temporary service certificate is created and installed in the local store. This certificate will be used to encrypt the message, protecting the username and password as well as any other sensitive data**.**

The security model for Hotline serivce has the following steps. These steps follows the implementation specified in [26]

- Step 1: Create a certificate to act as the root certificate authority

- Step 2: Create a certificate revocation list file from the root certificate
- Step 3: Install the root CA on the server and client machines
- Step 4: Install the certificate revocation list file on the server and client machines
- Step 5: Create and install the temporary service certificate
- Step 6: Give the WCF process identity access to the temporary certificate's private key

When developing a WCF service that uses X.509 certificates to provide message security, it is necessary to work with temporary certificates. This is because production certificates are expensive and may not be readily available. There are two options for specifying trust on a certificate

- **Peer trust** validates the certificate directly.
- **Chain trust** validates the certificate against the issuer of a certificate known as a root authority.
- The Hotline Service implements the chain trust option because it is the most commonly used approach in Business-to-Business (B2B) scenarios, and it is the default validation for WCF when using message security.
- To use chain trust validation during development time, I created a self-signed root certificate authority (CA) and installd it in the trusted root certification authority location in the local machine where the Hotline Service is developed. The certificate used by WCF is signed by the root self-signed certificate and installed in the personal store of the machine. To ensure that

- ‘makecert.exe’ is the tool used to create a private key file and a certificate to act as the root CA . The CRL file is then created from the private key that will act as the revocation list file for the root CA. Next, the root certificate and the CRL file are installed. Finally, create and install the temporary certificate from the root certificate, using the private key to sign and generate the key.

7.1 Create the certificate to act as the root certifying authority:

In this step, use the makecert tool to create a root CA that will be used to sign the certificate. This certificate will be self signed and will only have the **public key** that will be used to do the trust chain validation when encrypting and signing messages. The self-signed certificate will act as a root certificate itself, instead of pointing to a root authority in a chain of trust.

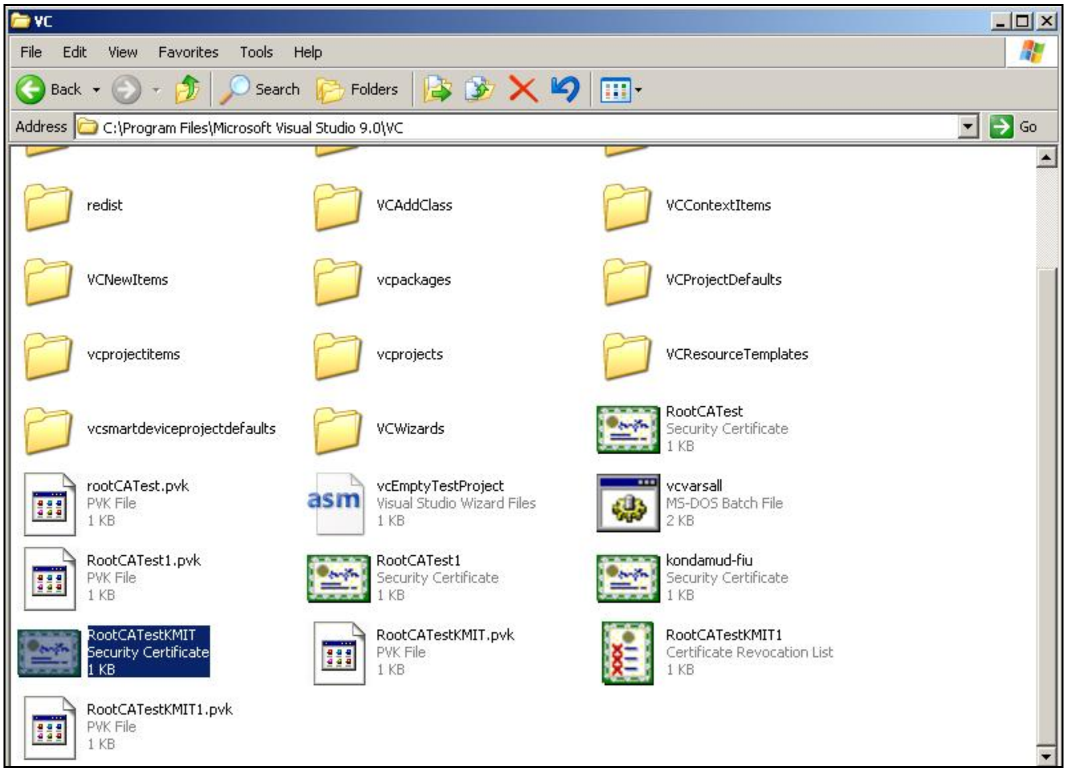
- Open a visual studio command prompt and browse to the location where to save the certificate files.
- Run the following command to create the root CA:
-
  ```
  makecert -n "CN=RootCATestKMIT” -r -sv RootCATestKMIT.pvk RootCATestKMIT.cer
  ```

In this command:

- **-n** specifies the subject name for the root CA. The convention is to prefix the subject name with "CN = " for "Common Name".
- **-r** specifies that the certificate will be self-signed. This means that certificates created with this switch will act as a root certificate.

- **-sv** specifies the file that will contain the private key of the certificate. The file is always created, if it does not exist. This will allow creating certificates using the private key file for signing and key generation.
- **RootCATestKMIT.cer** specifies the name of the file containing the public key of the certificate. The RootCATestKMIT.cer file will not have the private key. This is the certificate that will be installed in the store for trust chain validation on the client and server machines.
- In the **Create Private Key Password** dialog box, enter a password, confirm the password, and then click **OK**. Optionally, we can click **None** without entering the password, but this is not recommended for security reasons.
- In the **Enter Private Key Password** dialog box, enter the password again and then click **OK**.

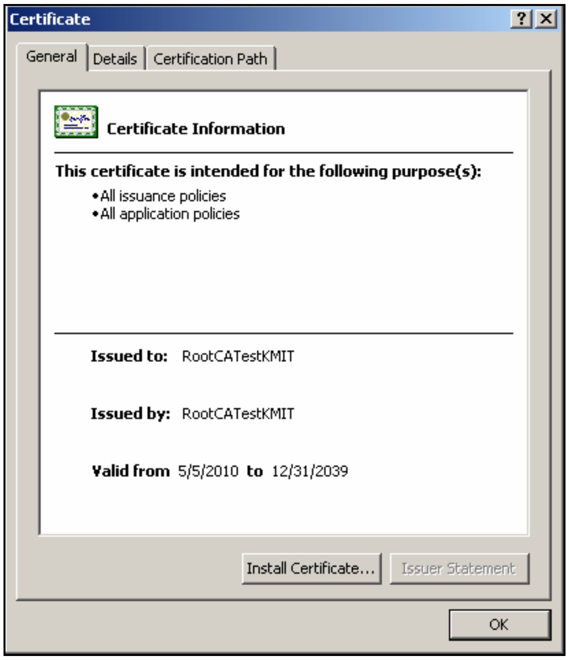
This is the password needed to access the private key file RootCATestKMIT.pvk in order to generate the file RootCATestKMIT.cer containing the public key.This step creates a certificate named RootCATestKMIT.cer and a private key file named RootCATestKMIT.pvk.We can browse to the location to confirm if the root certificate is created.



Address: C:\Program Files\Microsoft Visual Studio 9.0\VC

redist
VCAddClass
VCContextItems
VCNewItems
vcpackages
VCProjectDefaults
vcprojectitems
vcprojects
VCResourceTemplates
vcsmartdeviceprojectdefaults
VCWizards
RootCATest
Security Certificate
1 KB
rootCATest.pvk
PVK File
1 KB
vcEmptyTestProject
Visual Studio Wizard Files
1 KB
vcvarsall
MS-DOS Batch File
2 KB
RootCATest1.pvk
PVK File
1 KB
RootCATest1
Security Certificate
1 KB
kondamud-fiu
Security Certificate
1 KB
RootCATestKMIT
Security Certificate
1 KB
RootCATestKMIT.pvk
PVK File
1 KB
RootCATestKMIT1
Certificate Revocation List
1 KB
RootCATestKMIT1.pvk
PVK File
1 KB

Figure 14: Location showing the creation of RootCATestKMIT



Certificate Information

This certificate is intended for the following purpose(s):

- All issuance policies
- All application policies

Issued to: RootCATestKMIT

Issued by: RootCATestKMIT

Valid from 5/5/2010 **to** 12/31/2039

Figure 15: Certificate RootCATestKmit

Figures 14 and 15 display the location where the KMIT root certificate has been created and the display of the certificate respectively.

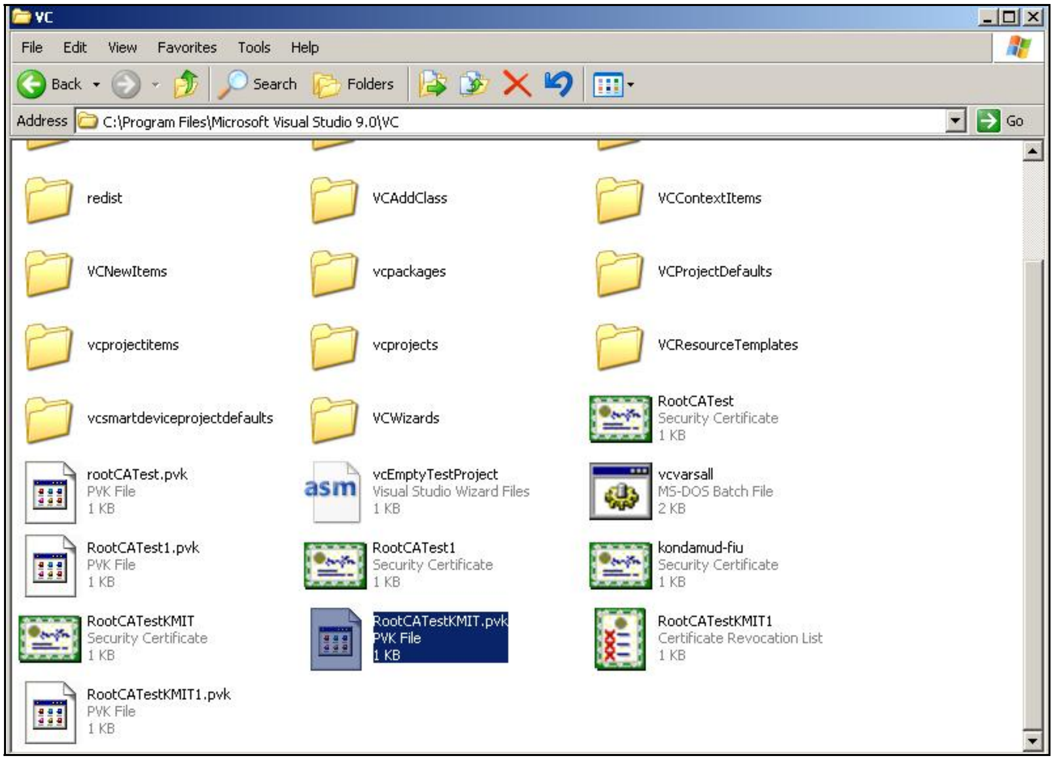
7.2 Create a certificate revocation list file from the root certificate

In this step, a CRL file is created that will be imported into the correct certificate stores of the client and service machines. This CRL is created for the temporary root certificate. The CRL is necessary because WCF clients check for the CRL when validating certificates .

- Open a visual studio command prompt and browse to the location where to save the CRL file for the root certificate.
- Run the following command to create the CRL file:
- `makecert -crl -n "CN=RootCATestKMIT" -r -sv RootCATestKMIT.pvk RootCATestKMIT.crl`

In this command:

- **-crl** specifies to generate the CRL file for the root certificate.
- **-n** specifies the subject name for the CRL. The convention is to prefix the subject name with "CN = " for "Common Name". It the same name as the root CA.
- **-r** specifies that the CRL file will be self-signed. This means that CRL files created with this switch will act as revocation list files for the root CA.
- **-sv** specifies the file that will contain the private key for CRL file generation. There is no need to create this file because it already exists. This will allow creation of CRL files using the private key file for signing.
- **RootCATestKMIT.crl** is the CRL file created with the command



Address: C:\Program Files\Microsoft Visual Studio 9.0\VC

redist
VCAddClass
VCContextItems
VCNewItems
vcpackages
VCProjectDefaults
vcprojectitems
vcprojects
VCResourceTemplates
vcsmartdeviceprojectdefaults
VCWizards
RootCATest Security Certificate 1 KB
rootCATest.pvk PVK File 1 KB
vcEmptyTestProject Visual Studio Wizard Files 1 KB
vcvarsall MS-DOS Batch File 2 KB
RootCATest1.pvk PVK File 1 KB
RootCATest1 Security Certificate 1 KB
kondamud-fiu Security Certificate 1 KB
RootCATestKMIT Security Certificate 1 KB
RootCATestKMIT.pvk PVK File 1 KB
RootCATestKMIT1 Certificate Revocation List 1 KB
RootCATestKMIT1.pvk PVK File 1 KB

Figure 16 : Location of RootCATestKMIT.pvk

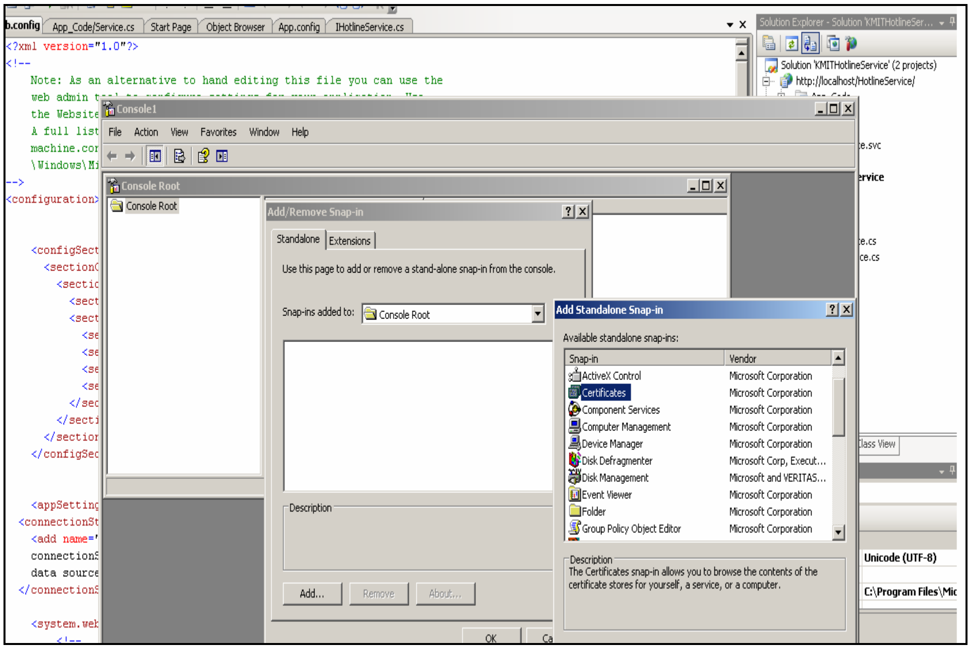
- The RootCATestKMIT cannot be read as it contains the private key, and service has the only authority to open the .pvk file.
- Figure 16 displays the location of the RootCATestKMIT.pvk file in the local machine.

7.3 Install the root certificate on client and server machines

This step explains how to install the certificate in the Trusted Root Certification Authorities location on both the server and client machines. Since while development of the Hotline Service the client and service are on same machine, it’s enough to install once. All certificates that are signed with this certificate will be trusted by the client machine**.**

- Copy the RootCATestKMIT.cer file to the client and server machines.

- Click **Start** and then click **Run**.
- In the command line, type **MMC** and then click **OK**.
- In the Microsoft Management Console, on the **File** menu, click **Add/Remove Snap-in**.
- In the **Add Remove Snap-in** dialog box, click **Add**.
- In the **Add Standalone Snap-in** dialog box, select **Certificates** and then click **Add**.



```
<?xml version="1.0"?>
<!--
    Note: As an alternative to hand editing this file you can use the
    web admin t
    the Website
    A full list
    machine.cor
    \Windows\Mi
-->
<configuration>

    <configSect
      <section(
        <sectic
          <sect
          <sect
            <se
            <se
            <se
            <se
          </sec
        </secti
      </sectior
    </configSec

    <appSetting
  <connectionSt
    <add name='
    connectionS
    data source
  </connectionS

    <system.wek
      <!--
```

Use this page to add or remove a stand-alone snap-in from the console.

Snap-ins added to: Console Root

Available standalone snap-ins:

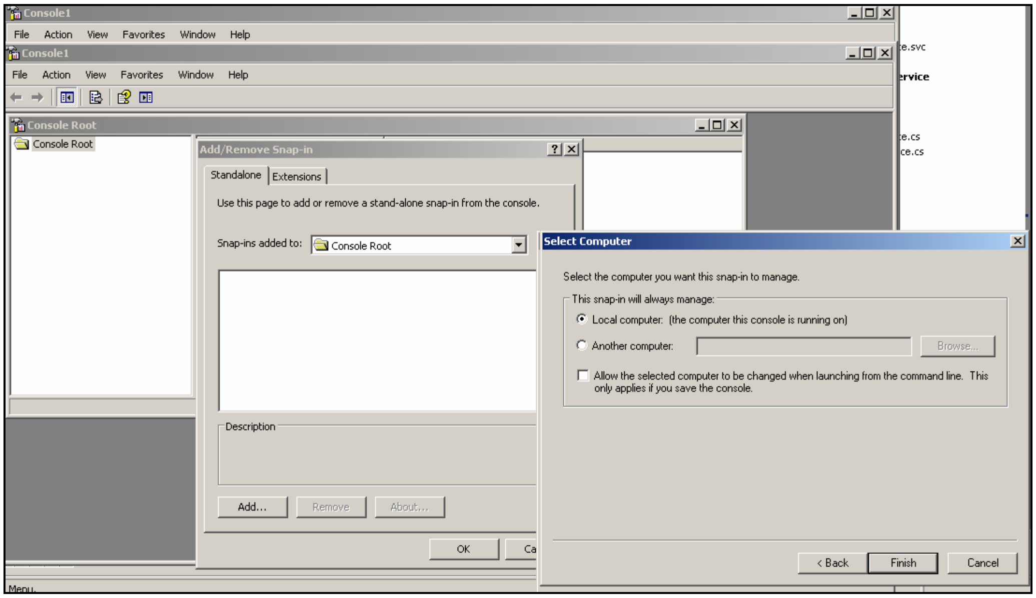
Snap-in	Vendor
ActiveX Control	Microsoft Corporation
Certificates	Microsoft Corporation
Component Services	Microsoft Corporation
Computer Management	Microsoft Corporation
Device Manager	Microsoft Corporation
Disk Defragmenter	Microsoft Corp, Execut...
Disk Management	Microsoft and VERITAS...
Event Viewer	Microsoft Corporation
Folder	Microsoft Corporation
Group Policy Object Editor	Microsoft Corporation

Description
The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.

Figure 17: Management console to add certificate

- In the **Certificates snap-in** dialog box, select the **Computer account** radio button because the certificate needs to be made available to all users, and then click **Next**.
- In the **Select Computer** dialog box, it is left to default option, the **Local computer: (the computer this console is running on)** is selected and then click **Finish**.

- In the **Add Standalone Snap-in** dialog box, click **Close**.



Use this page to add or remove a stand-alone snap-in from the console.

Snap-ins added to: Console Root

Select the computer you want this snap-in to manage.

This snap-in will always manage:

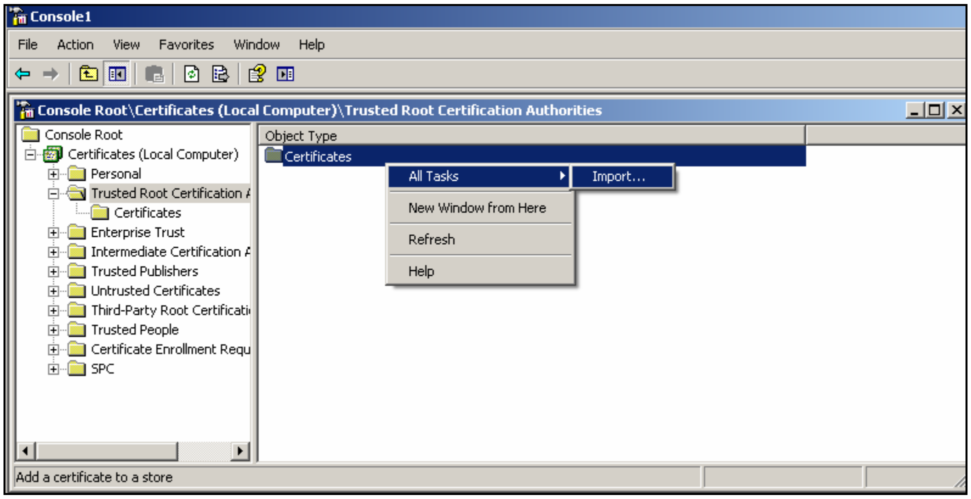
● Local computer: (the computer this console is running on)

○ Another computer:

□ Allow the selected computer to be changed when launching from the command line. This only applies if you save the console.

Figure 18: Add certificate to local computer

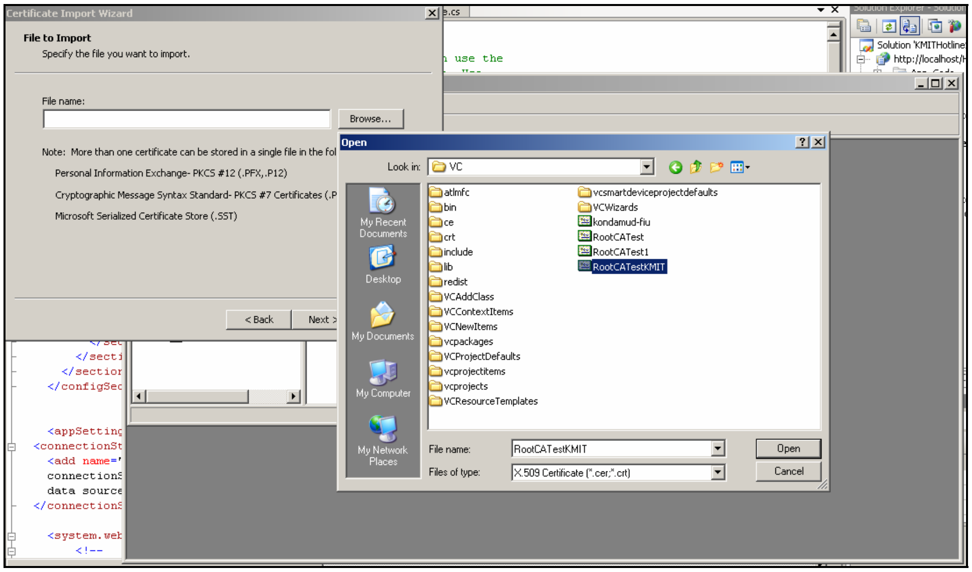
- In the **Add/Remove Snap-in** dialog box, click **OK**.
- In the left pane, expand the **Certificates (Local Computer)** node, and then expand the trusted root certification Authorities folder.
- Under **Trusted Root Certification Authorities**, right-click the **Certificates** subfolder, select **All Tasks**, and then click **Import**.


Console1
File Action View Favorites Window Help
Console Root\Certificates (Local Computer)\Trusted Root Certification Authorities
Console Root
Certificates (Local Computer)
Personal
Trusted Root Certification A
Certificates
Enterprise Trust
Intermediate Certification A
Trusted Publishers
Untrusted Certificates
Third-Party Root Certificati
Trusted People
Certificate Enrollment Requ
SPC
Object Type
Certificates
All Tasks
Import...
New Window from Here
Refresh
Help

Add a certificate to a store

Figure 19: Importing wizard for certificate into trusted root certification folder

- On the **Certificate Import Wizard** welcome screen, click **Next**.
- On the **File to Import** screen, click **Browse**.
- Browse to the location of the signed Root Certificate Authority RootCATestKMIT.cer file copied in Step 1, select the file, and then click **Open**.



Certificate Import Wizard

File to Import

Specify the file you want to import.

File name:

Note: More than one certificate can be stored in a single file in the fol

Personal Information Exchange- PKCS #12 (.PFX,.P12)

Cryptographic Message Syntax Standard- PKCS #7 Certificates (.P

Microsoft Serialized Certificate Store (.SST)

Open

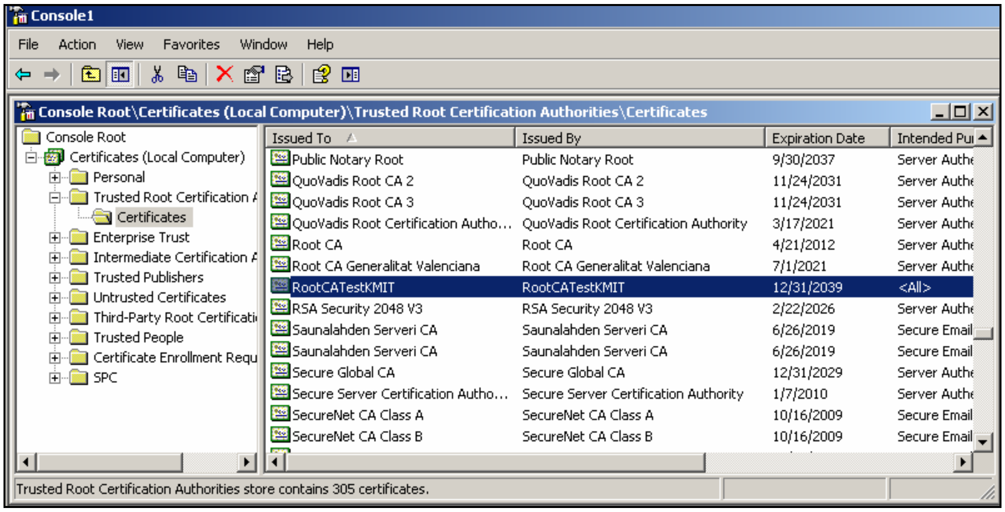
Look in: VC

File name: RootCATestKMIT

Files of type: X.509 Certificate (*.cer;*.crt)

Figure 20: Import wizard

- On the **File to Import** screen, click **Next**.
- On the **Certificate Store** screen, accept the default choice and then click **Next**.
- On the **Completing the Certificate Import Wizard** screen, click **Finish**.


Console1
File
Action
View
Favorites
Window
Help
Console Root\Certificates (Local Computer)\Trusted Root Certification Authorities\Certificates
Console Root
Certificates (Local Computer)
Personal
Trusted Root Certification A
Certificates
Enterprise Trust
Intermediate Certification A
Trusted Publishers
Untrusted Certificates
Third-Party Root Certificati
Trusted People
Certificate Enrollment Requ
SPC

Issued To	Issued By	Expiration Date	Intended Pu
Public Notary Root	Public Notary Root	9/30/2037	Server Authe
QuoVadis Root CA 2	QuoVadis Root CA 2	11/24/2031	Server Authe
QuoVadis Root CA 3	QuoVadis Root CA 3	11/24/2031	Server Authe
QuoVadis Root Certification Autho...	QuoVadis Root Certification Authority	3/17/2021	Server Authe
Root CA	Root CA	4/21/2012	Server Authe
Root CA Generalitat Valenciana	Root CA Generalitat Valenciana	7/1/2021	Server Authe
RootCATestKMIT	RootCATestKMIT	12/31/2039	<All>
RSA Security 2048 V3	RSA Security 2048 V3	2/22/2026	Server Authe
Saunalahden Serveri CA	Saunalahden Serveri CA	6/26/2019	Secure Email
Saunalahden Serveri CA	Saunalahden Serveri CA	6/26/2019	Secure Email
Secure Global CA	Secure Global CA	12/31/2029	Server Authe
Secure Server Certification Autho...	Secure Server Certification Authority	1/7/2010	Server Authe
SecureNet CA Class A	SecureNet CA Class A	10/16/2009	Secure Email
SecureNet CA Class B	SecureNet CA Class B	10/16/2009	Secure Email

Trusted Root Certification Authorities store contains 305 certificates.

Figure 21: Import wizard completed

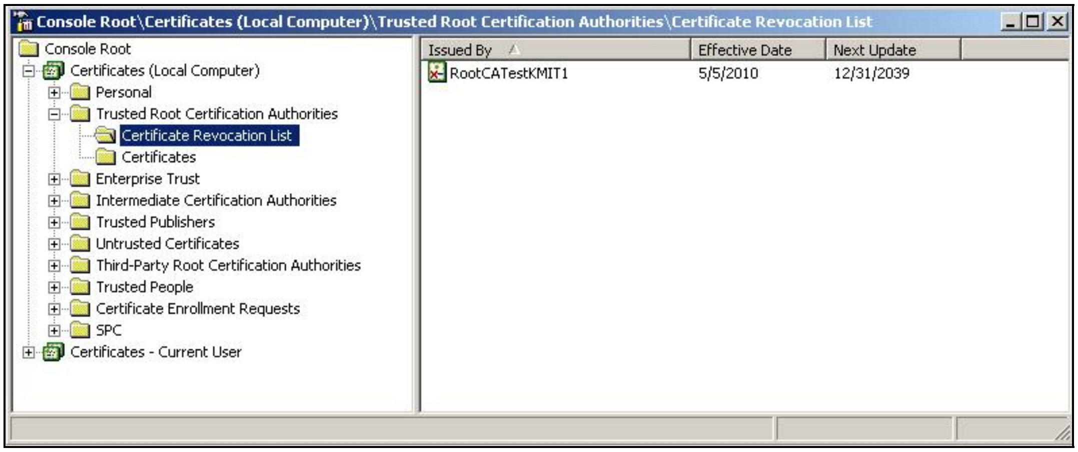
The signed root CA certificate is now installed in the trusted root certification authorities' store. You can expand the Certificates subfolder under trusted root certification authorities to see the 'RootCATestKMIT' certificate installed properly.

7.4 Install certificate revocation list file on client and server machines:

In this step, install the CRL from the file in the trusted root certification authorities' location on both the server and client machines. The CRL is checked during the certificate validation process**.**

- Copy the 'RootCATestKMIT.crl' file to the client and server machines.
- Click **Start** and then click **Run**.
- In the command line, type **MMC** and then click **OK**.
- In the Microsoft Management Console, on the **File** menu, click **Add/Remove Snap-in**.
- In the **Add Remove Snap-in** dialog box, click **Add**.

- In the **Add Standalone Snap-in** dialog box, select **Certificates** and then click **Add**.
- In the **Certificates snap-in** dialog box, select the **Computer account** radio button because the certificate needs to be made available to all users, and then click **Next**.
- In the **Select Computer** dialog box, leave the default **Local computer: (the computer this console is running on)** selected and then click **Finish**.
- In the **Add Standalone Snap-in** dialog box, click **Close**.
- In the **Add/Remove Snap-in** dialog box, click **OK**.
- In the left pane, expand the **Certificates (Local Computer)** node, and then expand the Trusted Root Certification Authorities folder.
- Under **Trusted Root Certification Authorities**, right-click the **Certificates** subfolder, select **All Tasks**, and then click **Import**.
- On the **Certificate Import Wizard** welcome screen, click **Next**.
- On the **File to Import** screen, click **Browse**.
- In **Files of Type**, select **Certificate Revocation List**.
- Browse to the location of the 'signed Root Certificate Authority' RootCATestKMIT1.crl file copied in Step 1, select the file, and then click **Open**.
- On the **File to Import** screen, click **Next**.
- On the **Certificate Store** screen, accept the default choice and then click **Next**.
- On the **Completing the Certificate Import Wizard** screen, click **Finish**.


Console Root\Certificates (Local Computer)\Trusted Root Certification Authorities\Certificate Revocation List
Console Root
Certificates (Local Computer)
Personal
Trusted Root Certification Authorities
Certificate Revocation List
Certificates
Enterprise Trust
Intermediate Certification Authorities
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification Authorities
Trusted People
Certificate Enrollment Requests
SPC
Certificates - Current User

Issued By	Effective Date	Next Update
RootCATestKMIT1	5/5/2010	12/31/2039

Figure 22: Certificate revocation list

The screenshots are similar to those done for installing root certificate on client and server machines. The CRL for the root CA certificate is now installed in the ‘trusted root certification authorities’ store. To view the CRL, click the trusted root certification authorities folder then press F5. A subfolder named certificate revocation list will be displayed. Expand this folder and you will see the ‘RootCATest CRL’ installed properly.

7.5 Create and install the temporary certificate

This step, describes the creation and installation of the temporary certificate on the server machine from the signed root CA created in the previous step.

- Open a visual studio command prompt and browse to the location where you have the ‘root CA’ certificate and private key file.

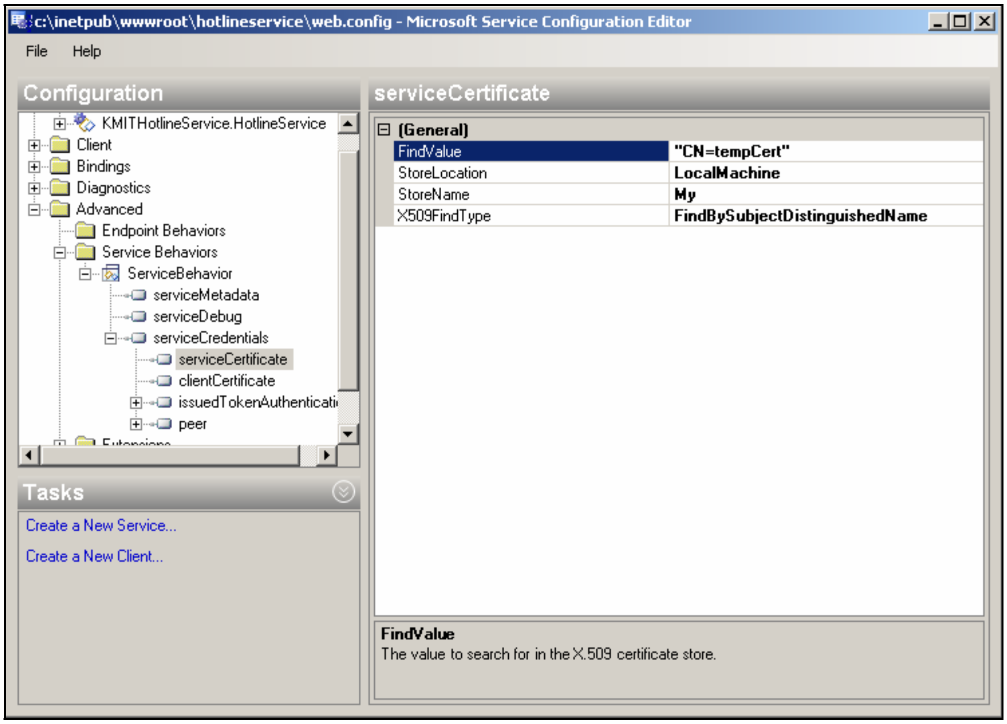
- Run following command for creating a certificate signed by the ‘root CA’ certificate:

-
  ```
  makecert -sk HotlineService -iv RootCATestKMIT.pvk -n "CN=tempCert"
  -ic RootCATestKMIT.cer -sr localmachine -ss my -sky exchange -pe
  ```

In this command:

- **-sk** specifies the key container name for the certificate. This needs to be unique for each certificate you create. Here ‘HotlineService’ is the name of the key container.
- **iv** specifies the private key file from which the temporary certificate will be created. You need to specify the root certificate private key file name that was created in the previous step and make sure that it is available in the current directory. This will be used for signing the certificate and for key generation.
- **-n** specifies the key subject name for the temporary certificate. The convention is to prefix the subject name with ‘CN = Common Name’.
- **-ic** specifies the file containing the root CA certificate file generated in the previous step.
- **-sr** specifies the store location where the certificate will be installed. The default location is Currentuser, but since the certificate needs to be available to all users, we should use the localmachine option.
- **-ss** specifies the store name for the certificate. **My** is the personal store location of the certificate.

- **-sky** specifies the key type, which could be either **signature** or **exchange**. Using **exchange** makes the certificate capable of signing and encrypting the message.
- **-pe** specifies that the private key is generated in the certificate and installed with it in the certificate store. When we double-click the certificate, on the **General** tab, it displays a message at the bottom stating, "**You have a private key that corresponds to this certificate**". This is a requirement for message security. If the certificate does not have the corresponding private key, it cannot be used for message security.



FindValue	"CN=tempCert"
StoreLocation	LocalMachine
StoreName	My
X509FindType	FindBySubjectDistinguishedName

FindValue
The value to search for in the X.509 certificate store.

Figure 23: Service temporary certificate

- In the **Enter Private Key Password** dialog box, enter the password for the root CA private key file specified in Step 2, and then click **OK**.

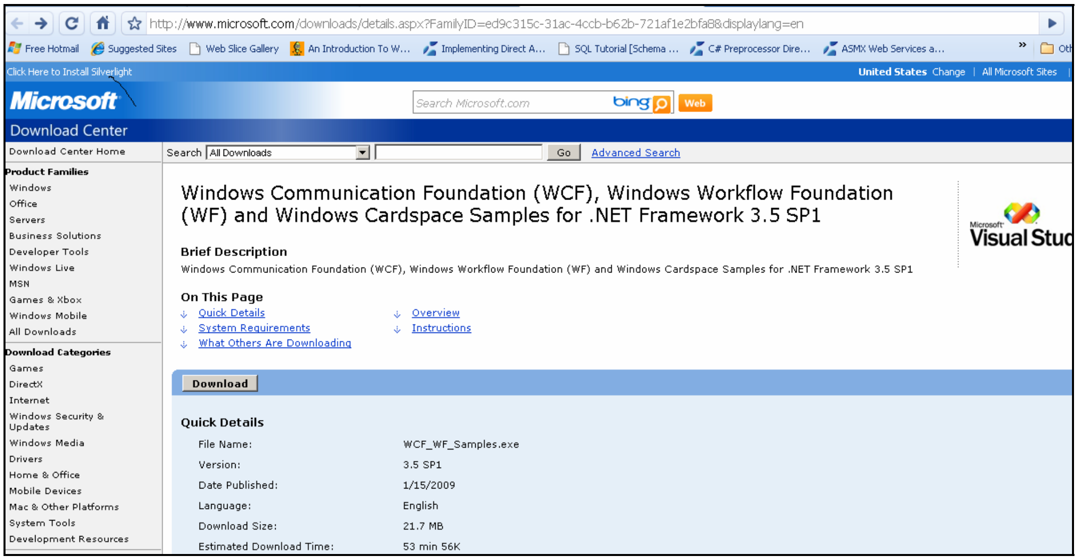
7.6 Give WCF process identity access to temporary certificate's private key:

In this step, you give the process identity of the WCF service access permissions to the certificate's private key. The Hotline Service is hosted in Internet Information Services (IIS), the identity typically is "ASPNET"; in a production scenario, or if your service is hosted in a windows service, it could be a custom domain service account.

- Open a visual studio command prompt.
- Run the following command:
- `FindPrivateKey.exe My LocalMachine -n "CN=tempCert"`

In this command:

- **My** is the store name where the temporary certificate is installed
- **LocalMachine** is the store location for the certificate.
- **–n "CN=tempCert"** is the common name for the temporary certificate.
- If 'FindPrivateKey' is not on your machine, download the WCF samples, including the FindPrivateKey *http://www.microsoft.com/downloads/details.aspx?FamilyId=2611A6FF-FD2D-4F5B-A672-C002F1C09CCD&displaylang=en*



Windows Communication Foundation (WCF), Windows Workflow Foundation (WF) and Windows Cardspace Samples for .NET Framework 3.5 SP1

Brief Description

Windows Communication Foundation (WCF), Windows Workflow Foundation (WF) and Windows Cardspace Samples for .NET Framework 3.5 SP1

Quick Details

File Name:	WCF_WF_Samples.exe
Version:	3.5 SP1
Date Published:	1/15/2009
Language:	English
Download Size:	21.7 MB
Estimated Download Time:	53 min 56K

Figure 24: Downloading private key tool

- ‘FindPrivateKey’ returns the location of the private key for the certificate as C:\Documents and Settings\All users\ApplicationData\Microsoft\Crypto\RSA\MachineKeys\9cb42596a80376b68a69f1ff169bd226_e117b178-9b4b-4f5d-a7c3-e039cc793450
- Run the following command to assign access permissions to the process identity of the WCF service.

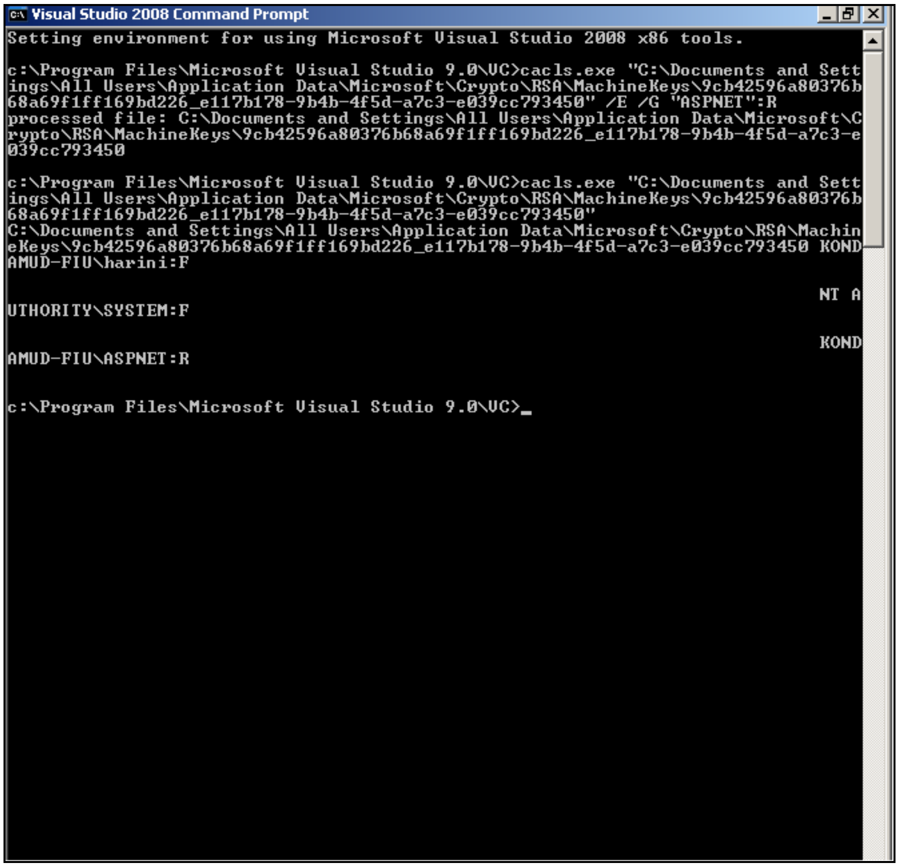
```
cacls.exe "C:\Documents and Settings\All Users\Application Data\Microsoft\Crypto\RSA\Machinekeys\9cb42596a80376b68a69f1ff169bd226_e117b178-9b4b-4f5d-a7c3-e039cc793450" /E /G "ASPNET":R
```

In this command:

- **/E** edits the access control list (ACL) of the private key instead or replacing it. The ACL should never be replaced but should only add the necessary permission to the process identity.

- **/G** grants the permission to the process identity.
- **: R** gives read-only permissions to "ASPNET".
- Run the following command to verify the permissions on the private key. This will display all the identities and the permissions that have access to the private key
- `acls.exe "C:\Documents and Settings\All Users\Application Data\Microsoft\Crypto\RSA\Machinekeys\9cb42596a80376b68a69f1ff169bd226_e117b178-9b4b-4f5d-a7c3-e039cc793450`

  We should see the following in the output from this command: `ASPNET: R`



Visual Studio 2008 Command Prompt

```
Setting environment for using Microsoft Visual Studio 2008 x86 tools.

c:\Program Files\Microsoft Visual Studio 9.0\VC>cacls.exe "C:\Documents and Sett
ings\All Users\Application Data\Microsoft\Crypto\RSA\MachineKeys\9cb42596a80376b
68a69f1ff169bd226_e117b178-9b4b-4f5d-a7c3-e039cc793450" /E /G "ASPNET":R
processed file: C:\Documents and Settings\All Users\Application Data\Microsoft\C
rypto\RSA\MachineKeys\9cb42596a80376b68a69f1ff169bd226_e117b178-9b4b-4f5d-a7c3-e
039cc793450

c:\Program Files\Microsoft Visual Studio 9.0\VC>cacls.exe "C:\Documents and Sett
ings\All Users\Application Data\Microsoft\Crypto\RSA\MachineKeys\9cb42596a80376b
68a69f1ff169bd226_e117b178-9b4b-4f5d-a7c3-e039cc793450"
C:\Documents and Settings\All Users\Application Data\Microsoft\Crypto\RSA\Machin
eKeys\9cb42596a80376b68a69f1ff169bd226_e117b178-9b4b-4f5d-a7c3-e039cc793450 KOND
AMUD-FIU\harini:F

                                                                            NT A
UTHORITY\SYSTEM:F

                                                                            KOND
AMUD-FIU\ASPNET:R


c:\Program Files\Microsoft Visual Studio 9.0\VC>_
```

Figure 25 : Temporary key for service

It should be noted that temporary certificates should only be used for development and testing purposes. For real-world production environments, use a certificate provided by a CA such as Microsoft Windows Server® 2003 Certificate Services or a third party. This describes the ws-message security model implementation in WCF for Hotline Service as following the WS-Security protocol detailed in chapter-5

8.0 HOSTING ON IIS

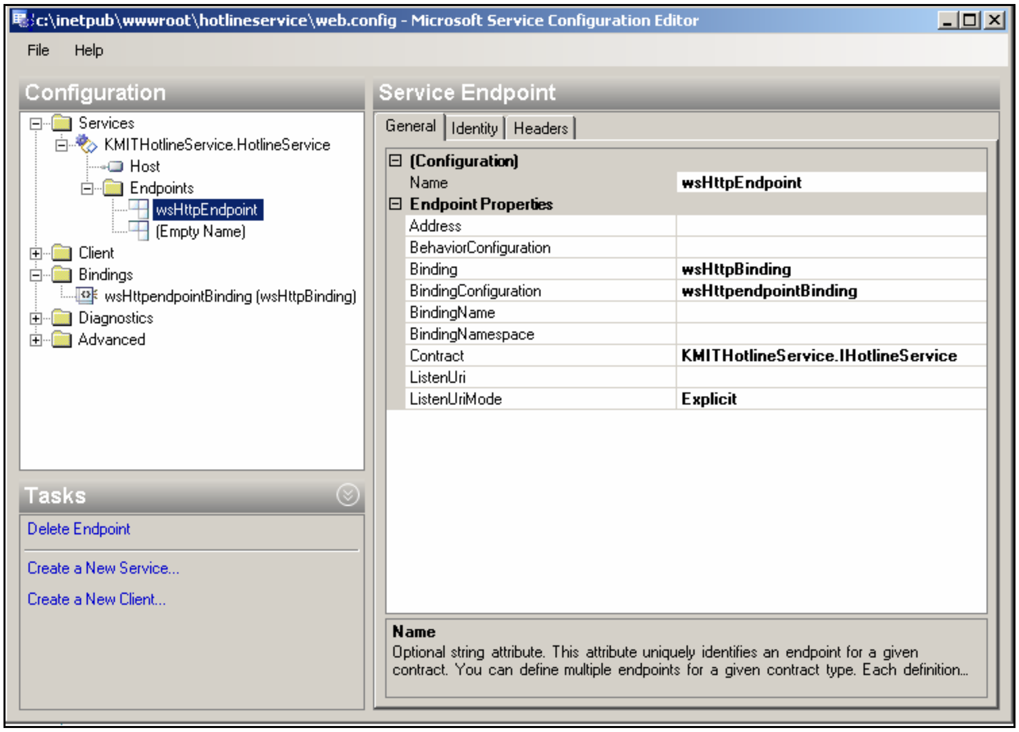
Chapter- 3 discusses the various mechanisms to be used for hosting services in WCF. The unified programming model of WCF is based on a strictly layered model to break the web-oriented paradigm and disconnect the service model and channel layer from the supported transports. This model allows WCF to support several different hosts of which IIS is the most important. Sites are bound to a particular scheme, network address, and port combination. IIS not only supports HTTP but also, depending on the version, FTP, NNTP, and SMTP. It can run multiple applications under the same site and under the same scheme, network, and port combination. A typical URI for an application is **http://localhost/MyApplication**. A virtual directory is simply a folder that is mapped to the network space of the site, which could be somewhere else on the file system. This way, developers can keep the actual content or code of an application separate from the other applications that are part of the same site.

The steps to host the Hotline Service on IIS are:

- Open visual studio solution explorer and open a new website in the same solution explorer and point the file location to **http://localhost/HotlineService**
- This project consist a HotlineService.svc file which needs to be configured to map to Hotline Service. Open HotlineService.svc file and change the code to

```
ServiceHost                Language="C#"               Debug="true"
Service="KMITHotlineService.HotlineService"
```

- Configure the website with some endpoints. Open edit WCF configuration and re configure the endpoints to use IHOTLINESERVICE endpoint. Set the address, binding and contract information that enables the client to point to IHotlineService. Save the configuration file and exut out of the tool.

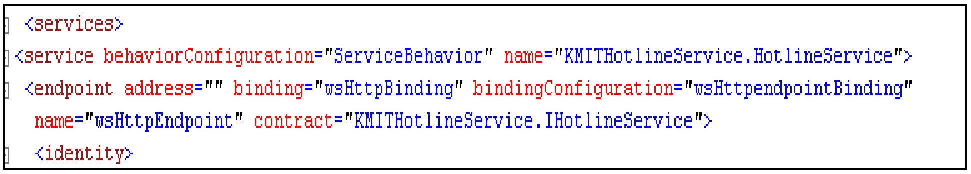


(Configuration)	
Name	wsHttpEndpoint
Endpoint Properties	
Address	
BehaviorConfiguration	
Binding	wsHttpBinding
BindingConfiguration	wsHttpendpointBinding
BindingName	
BindingNamespace	
Contract	KMITHotlineService.IHotlineService
ListenUri	
ListenUriMode	Explicit

Name
Optional string attribute. This attribute uniquely identifies an endpoint for a given contract. You can define multiple endpoints for a given contract type. Each definition...

Figure 26: WCF configuration editor

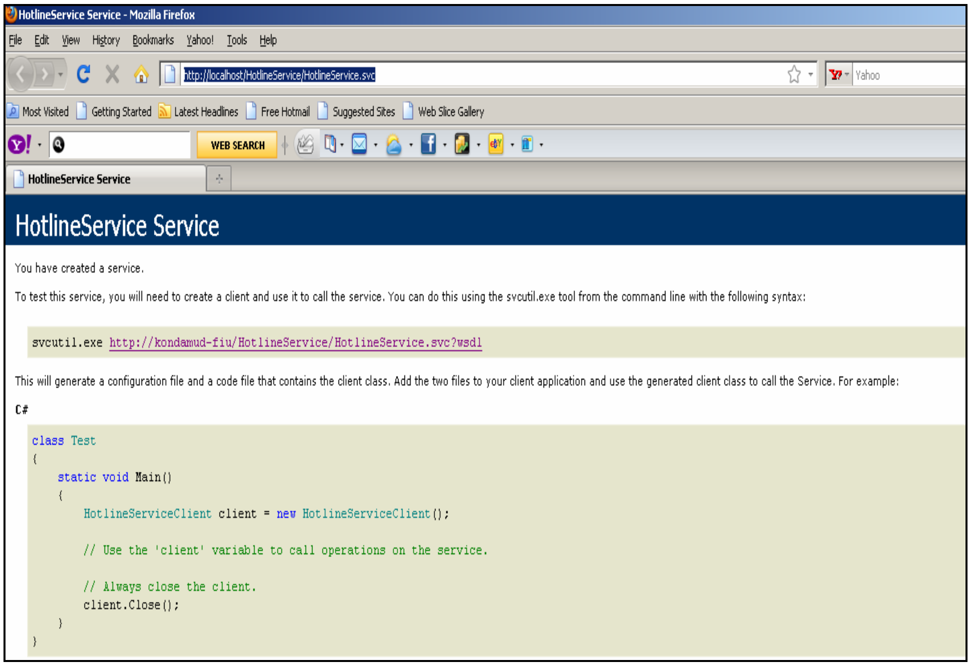
- The code from the web.config file looks similar to this.



```
<services>
<service behaviorConfiguration="ServiceBehavior" name="KMITHotlineService.HotlineService">
 <endpoint address="" binding="wsHttpBinding" bindingConfiguration="wsHttpendpointBinding"
  name="wsHttpEndpoint" contract="KMITHotlineService.IHotlineService">
  <identity>
```

Figure 27: Web.config file

- When we right click the HotlineService.svc file and browse, it loads the Hotline Service at address 'http://localhost/HotlineService/HotlineService.svc'. This address specifies that the service is hosted on IIS.



HotlineService Service - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://localhost/HotlineService/HotlineService.svc
Yahoo
Most Visited Getting Started Latest Headlines Free Hotmail Suggested Sites Web Slice Gallery
WEB SEARCH
HotlineService Service

HotlineService Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
svcutil.exe http://kondamud-fiu/HotlineService/HotlineService.svc?wsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

C#

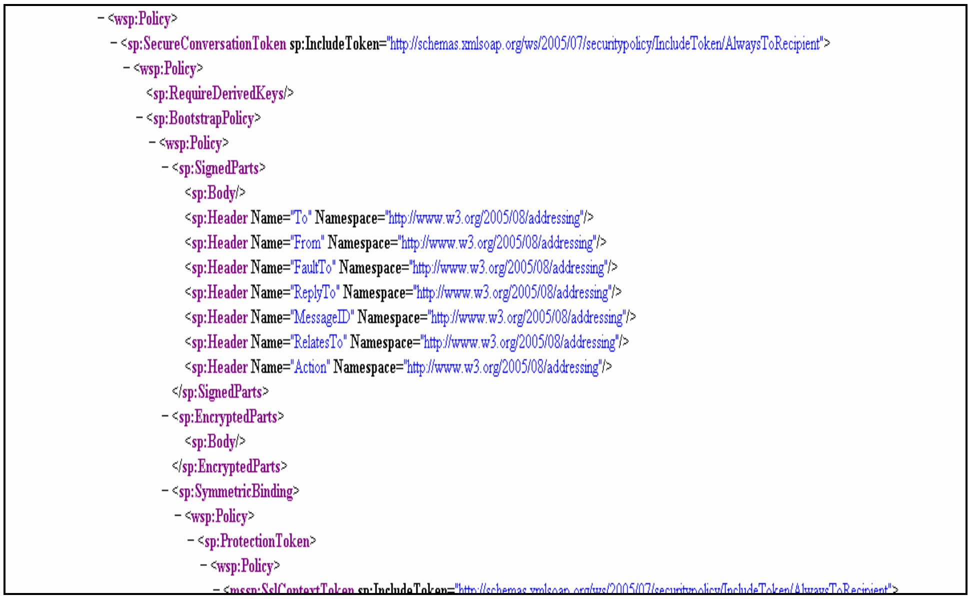
```
class Test
{
    static void Main()
    {
        HotlineServiceClient client = new HotlineServiceClient();

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

Figure 28: Hotline service on IIS

- Click the 'service.util' to expose the WSDL of the service.



```
- <wsp:Policy>
  - <sp:SecureConversationToken sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
    - <wsp:Policy>
        <sp:RequireDerivedKeys/>
      - <sp:BootstrapPolicy>
        - <wsp:Policy>
          - <sp:SignedParts>
              <sp:Body/>
              <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
              <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
              <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
              <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
              <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
              <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
              <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
            </sp:SignedParts>
          - <sp:EncryptedParts>
              <sp:Body/>
            </sp:EncryptedParts>
          - <sp:SymmetricBinding>
            - <wsp:Policy>
              - <sp:ProtectionToken>
                - <wsp:Policy>
                  - <mssp:SslContextToken sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
```

Figure 29: WSDL for Hotline Service

9.0 CLIENTS TO CONSUME HOTLINESERVICE

9.1 Create a test client:

In this step, you create a windows forms application to test the WCF service.

- Right-click your solution, click Add, and then click New Project.
- In the Add New Project dialog box, in the Templates section, select Windows Forms Application.
- In the Name field, type HotlineClient_test and then click OK**.**
- This opens a 'forms' project in the same solution explorer which acts as a client to the Hotline Service

9.2 Add a web reference to the client

In this step, we add a reference to the Hotline Service.

- Right-click HotlineClient_test project and select **Add Web Reference**.
- In the **Add Web Reference** dialog box, set the URL to your WCF service (e.g., http://localhost/WCFTestService/Service.svc) and then click **Go**.
- In the **Web reference name** field, change ServiceReference1 to **HotlineClient_Test_Reference**
- Click **Add Reference**.

In the Client project, a reference to WCFTestService should now appear beneath **Service References**.

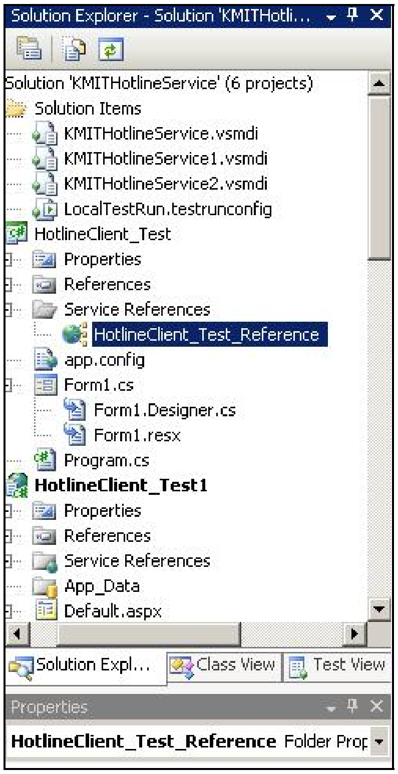


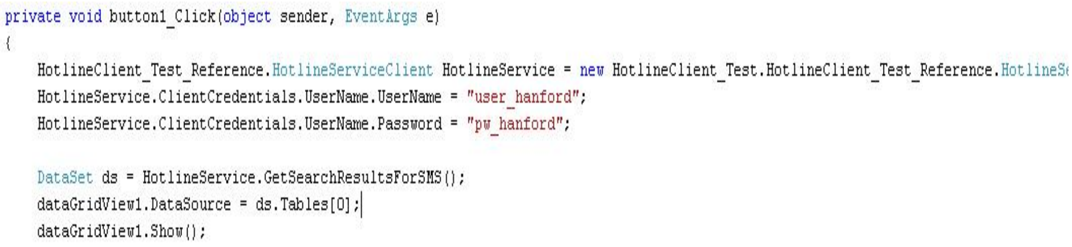
Figure 30: Test client in service reference

9.3 Test the client and WCF service

In this step, to access the WCF service, pass the user credentials, and make sure that the username authentication works.

- In your Client project, drag a button control onto the form, and name it as GetSearchResults
- Double-click the button control to show the underlying code.

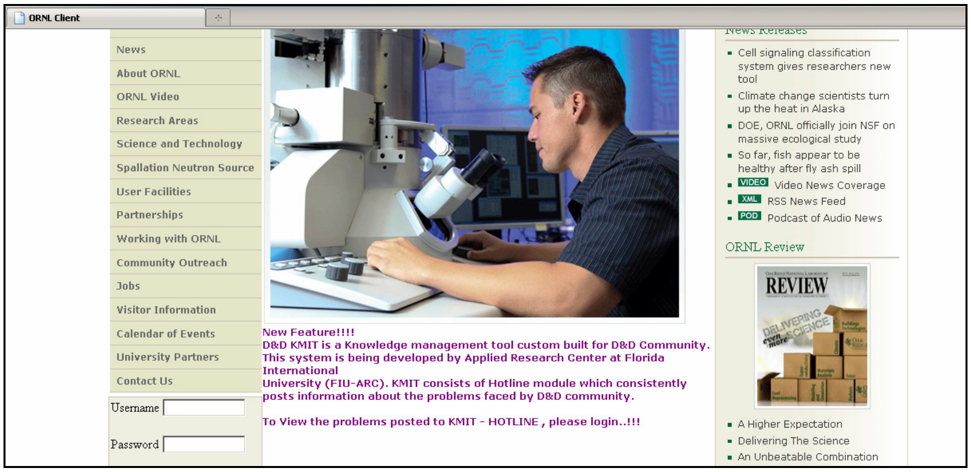
- Create an instance of the proxy, pass the credentials of the user and then call the operation of your WCF service. The code should look as follows



```
private void button1_Click(object sender, EventArgs e)
{
    HotlineClient_Test_Reference.HotlineServiceClient HotlineService = new HotlineClient_Test.HotlineClient_Test_Reference.HotlineS
    HotlineService.ClientCredentials.UserName.UserName = "user_hanford";
    HotlineService.ClientCredentials.UserName.Password = "pw_hanford";

    DataSet ds = HotlineService.GetSearchResultsForSMS();
    dataGridView1.DataSource = ds.Tables[0];
    dataGridView1.Show();
```

- Right-click the client project and select set as ‘startup project'.
- Run the client application by pressing F5 or CTRL+F5.



Username

Password

New Feature!!!!
D&D KMIT is a Knowledge management tool custom built for D&D Community. This system is being developed by Applied Research Center at Florida International
University (FIU-ARC). KMIT consists of Hotline module which consistently posts information about the problems faced by D&D community.

To View the problems posted to KMIT - HOTLINE , please login..!!!

- Cell signaling classification system gives researchers new tool
- Climate change scientists turn up the heat in Alaska
- DOE, ORNL officially join NSF on massive ecological study
- So far, fish appear to be healthy after fly ash spill
- VIDEO Video News Coverage
- XML RSS News Feed
- POD Podcast of Audio News

- A Higher Expectation
- Delivering The Science
- An Unbeatable Combination

Figure 31: Client application

Figure 31 shows a client application (ORNL) which is integrated with the Hotline Service and has a login control, which allows clients to login to access the service information.

10.0 FIDDLER:

Fiddler is a web debugging proxy which logs all HTTP(S) traffic between the computer and the internet. Fiddler allows inspecting all HTTP (S) traffic, setting breakpoints, and “fiddling" with incoming or outgoing data. Fiddler includes a powerful event-based scripting subsystem, and can be extended using any .NET language. Fiddler is a freeware and can debug traffic from virtually any application, including Internet Explorer, Mozilla Firefox, Opera, and others. Fiddler is a transparent proxy that automatically adds itself to the WININET chain so that it can see every request being made. It logs those requests and the responses to allow the application to see what is working and what isn't working.

Currently, as the Hotline Service was developed, and hosted, a client is created to access the Hotline Service. In order to verify if everything is going properly, we use Fiddler and intercept the traffic from the client to the Hotline Service. We try to get data from the fiddler and see if the service is correctly enabled with ws-message security. We can also inspect the binary security token that is used for communication between client and service, the way it is encrypted, algorithms used for encryption and the cipher text generated. However, we cannot decipher the cipher text, as the private key is only available with the service and service is the only application which is able to decrypt the user credentials, and allow them to access the information.

As the client authenticates the server using the ‘username/password’ credential, we first try to fiddle the data by giving the right credentials assigned to the client application. Then we try to give the wrong password and again try to fiddle the data to note the changes. Similar kind of testing is done with the service authentication. The user is first provided with correct certificate, and then the certificate is changed to show how message security is working with Hotline Service.

10.1 Experimental result - 1:

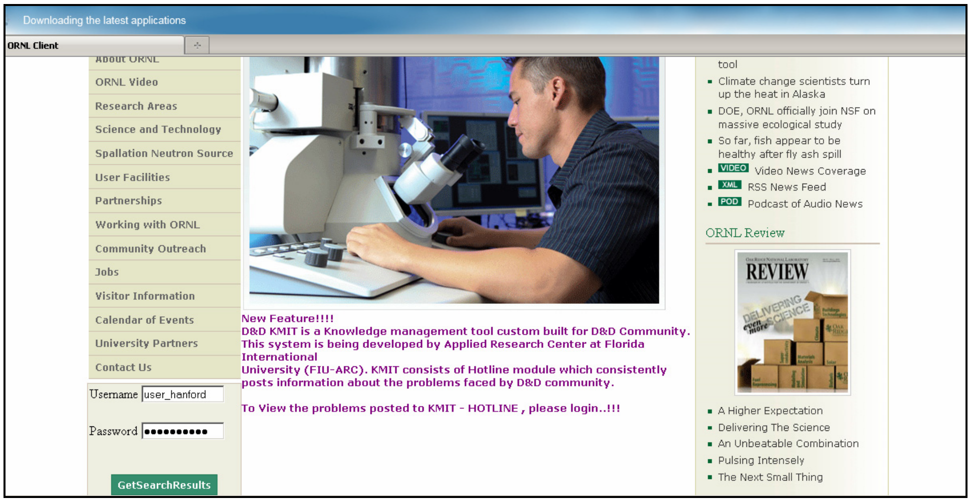
This is a best case scenario in which the client gives genuine credentials to access the service. Every client application has a specific username, password pair to access the service, which is stored in the database at the service end.

Client Credentials: username: user_hanford ; password : pw_hanford

Client: Oak Ridge National Laboratory application

Service Certificate: tempCert signed by RootCA

Expected output: On entering the username and password and clicking the button ‘GetSearchResults’, the client application should talk to the Web Service and display the top five problems of KMIT in ORNL application. When the client hits the service, the service should validate the client credentials against the entries in the database, and grant access only if it’s a genuine application. Fiddler tracks all this information and displays which is shown as a screenshot.



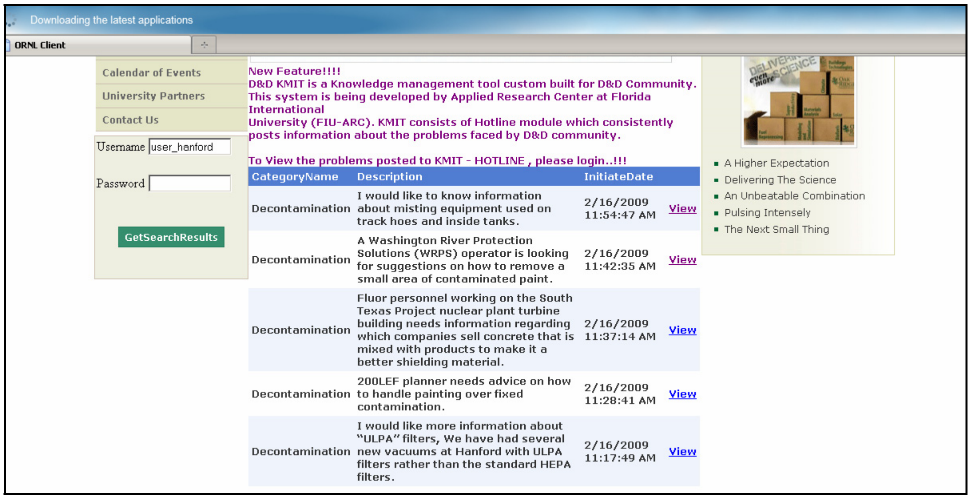
Username user_hanford

Password ●●●●●●●●●●●

New Feature!!!!
D&D KMIT is a Knowledge management tool custom built for D&D Community. This system is being developed by Applied Research Center at Florida International
University (FIU-ARC). KMIT consists of Hotline module which consistently posts information about the problems faced by D&D community.

To View the problems posted to KMIT - HOTLINE , please login..!!!

Figure 32: Client application with credentials



Username user_hanford

Password

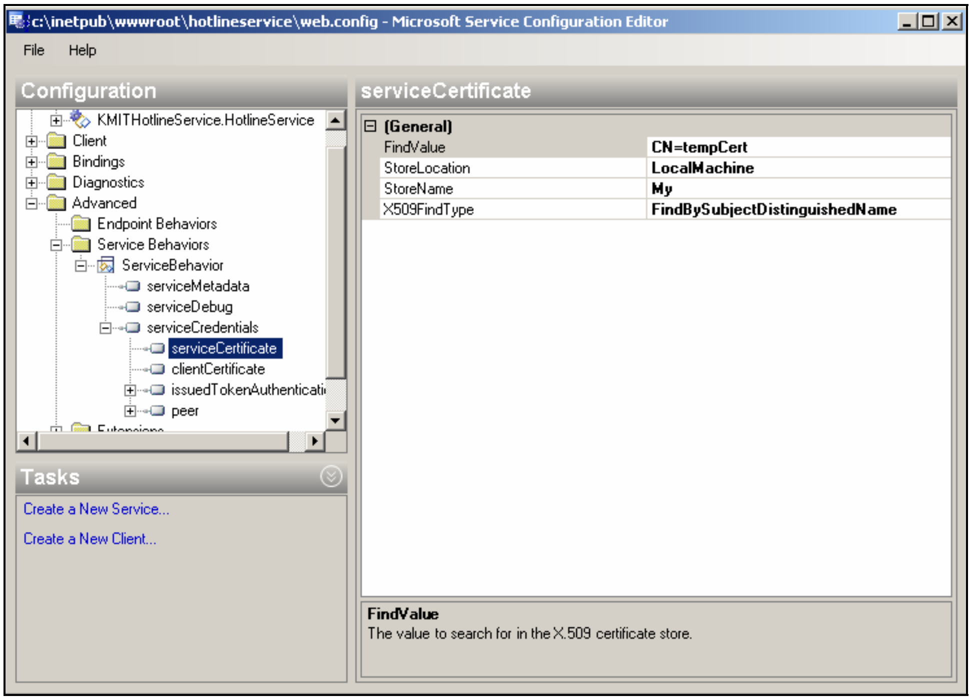
New Feature!!!!
D&D KMIT is a Knowledge management tool custom built for D&D Community. This system is being developed by Applied Research Center at Florida International
University (FIU-ARC). KMIT consists of Hotline module which consistently posts information about the problems faced by D&D community.

To View the problems posted to KMIT - HOTLINE , please login..!!!

CategoryName	Description	InitiateDate	
Decontamination	I would like to know information about misting equipment used on track hoes and inside tanks.	2/16/2009 11:54:47 AM	View
Decontamination	A Washington River Protection Solutions (WRPS) operator is looking for suggestions on how to remove a small area of contaminated paint.	2/16/2009 11:42:35 AM	View
Decontamination	Fluor personnel working on the South Texas Project nuclear plant turbine building needs information regarding which companies sell concrete that is mixed with products to make it a better shielding material.	2/16/2009 11:37:14 AM	View
Decontamination	200LEF planner needs advice on how to handle painting over fixed contamination.	2/16/2009 11:28:41 AM	View
Decontamination	I would like more information about "ULPA" filters, We have had several new vacuums at Hanford with ULPA filters rather than the standard HEPA filters.	2/16/2009 11:17:49 AM	View

Figure 33: Client application successfully displays results from KMIT

Background process/ Discussion: On button click, the client application should generate a *binary security token* from the credentials for communication. The security token is *encoded* using an encryption algorithm. The public key generated from the service certificate is used for encryption. The service gets started and starts listening for a request. When the request hits the service, it picks the encrypted data, uses its own private key pair to decrypt the information, and checks the information against the value in the database. If that is validated the client is allowed to access the service.



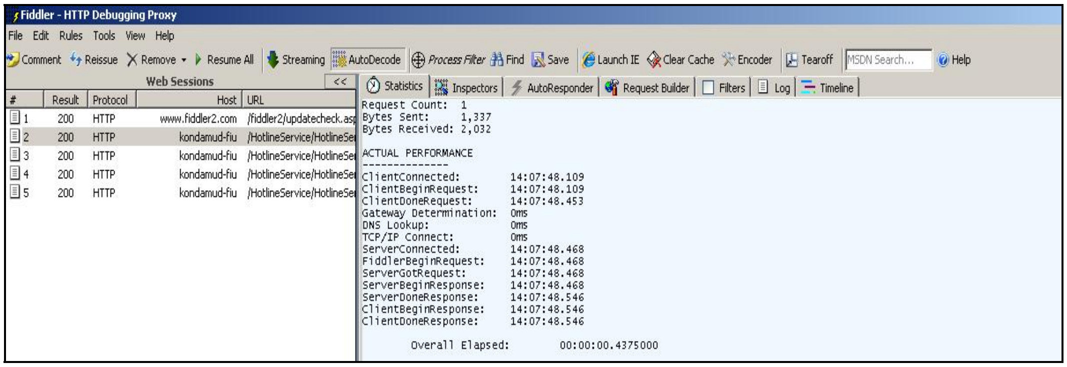
(General)

FindValue	CN=tempCert
StoreLocation	LocalMachine
StoreName	My
X509FindType	FindBySubjectDistinguishedName

FindValue
The value to search for in the X.509 certificate store.

Figure 34: Service referenced with its certificate tempCert

The following snapshots show what happens when a fiddler is active and intercepts the traffic


Fiddler - HTTP Debugging Proxy
File Edit Rules Tools View Help
Comment Reissue Remove Resume All Streaming AutoDecode Process Filter Find Save Launch IE Clear Cache Encoder Tearoff MSDN Search... Help
Web Sessions
<<
Statistics Inspectors AutoResponder Request Builder Filters Log Timeline

#	Result	Protocol	Host	URL
1	200	HTTP	www.fiddler2.com	/fiddler2/updatecheck.asp
2	200	HTTP	kondamud-fiu	/HotlineService/HotlineSer
3	200	HTTP	kondamud-fiu	/HotlineService/HotlineSer
4	200	HTTP	kondamud-fiu	/HotlineService/HotlineSer
5	200	HTTP	kondamud-fiu	/HotlineService/HotlineSer

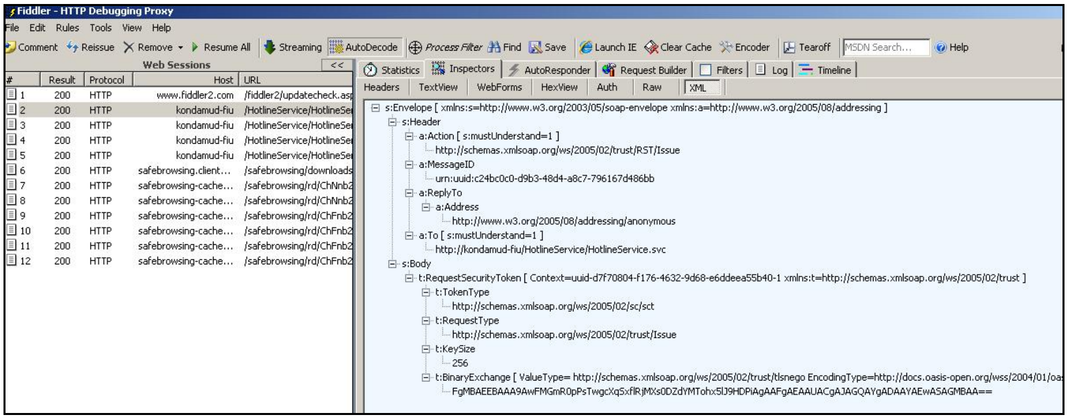
```
Request Count:  1
Bytes Sent:     1,337
Bytes Received: 2,032

ACTUAL PERFORMANCE
--------------
ClientConnected:        14:07:48.109
ClientBeginRequest:     14:07:48.109
ClientDoneRequest:      14:07:48.453
Gateway Determination:  0ms
DNS Lookup:             0ms
TCP/IP Connect:         0ms
ServerConnected:        14:07:48.468
FiddlerBeginRequest:    14:07:48.468
ServerGotRequest:       14:07:48.468
ServerBeginResponse:    14:07:48.468
ServerDoneResponse:     14:07:48.546
ClientBeginResponse:    14:07:48.546
ClientDoneResponse:     14:07:48.546

        Overall Elapsed:        00:00:00.4375000
```

Figure 35: Fiddler capture for statistics

The figure 32 displays the client application in which the credentials are passed in the login control. Figure 33 displays that the client application has successfully performed the handshake with the service and displayed the top five problems of KMIT in its own application. This explains the integration of a piece of software with another, which runs on different frameworks. Figure 34 shows the service referenced with its certificate tempCert signed by RootCA. Figure 35 shows the interface of fiddler, which displays the statistics generated when the client has started hitting the server. It shows the series of operations invoked beginning from the client generating request to the last action of client done with response. The overall statistics show that the communication between the client and the service is successful and the client is ok with the response that is given from the service. Furthur screenshots explain more about the details.

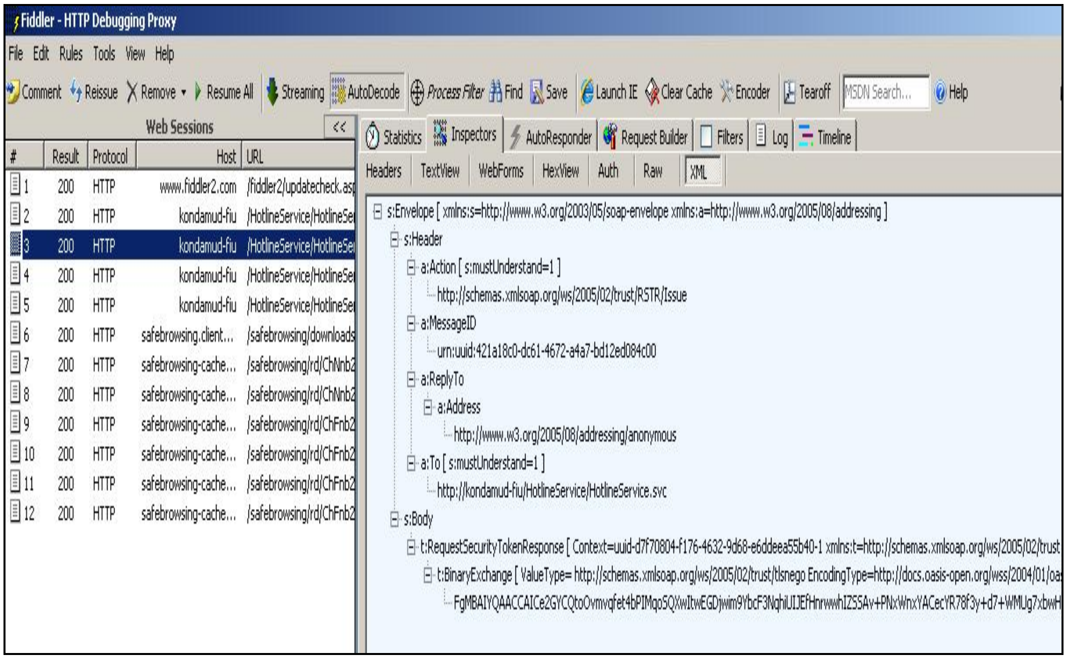

Fiddler - HTTP Debugging Proxy
File Edit Rules Tools View Help
Comment Reissue Remove Resume All Streaming AutoDecode Process Filter Find Save Launch IE Clear Cache Encoder Tearoff MSDN Search... Help
Web Sessions
<<
Statistics Inspectors AutoResponder Request Builder Filters Log Timeline
Headers TextView WebForms HexView Auth Raw XML

#	Result	Protocol	Host	URL
1	200	HTTP	www.fiddler2.com	/fiddler2/updatecheck.asp
2	200	HTTP	kondamud-fiu	/HotlineService/HotlineSe
3	200	HTTP	kondamud-fiu	/HotlineService/HotlineSe
4	200	HTTP	kondamud-fiu	/HotlineService/HotlineSe
5	200	HTTP	kondamud-fiu	/HotlineService/HotlineSe
6	200	HTTP	safebrowsing.client...	/safebrowsing/downloads
7	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChNnb2
8	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChNnb2
9	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
10	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
11	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
12	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2

```
s:Envelope [ xmlns:s=http://www.w3.org/2003/05/soap-envelope xmlns:a=http://www.w3.org/2005/08/addressing ]
  s:Header
    a:Action [ s:mustUnderstand=1 ]
      http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue
    a:MessageID
      urn:uuid:c24bc0c0-d9b3-48d4-a8c7-796167d486bb
    a:ReplyTo
      a:Address
        http://www.w3.org/2005/08/addressing/anonymous
    a:To [ s:mustUnderstand=1 ]
      http://kondamud-fiu/HotlineService/HotlineService.svc
  s:Body
    t:RequestSecurityToken [ Context=uuid-d7f70804-f176-4632-9d68-e6ddeea55b40-1 xmlns:t=http://schemas.xmlsoap.org/ws/2005/02/trust ]
      t:TokenType
        http://schemas.xmlsoap.org/ws/2005/02/sc/sct
      t:RequestType
        http://schemas.xmlsoap.org/ws/2005/02/trust/Issue
      t:KeySize
        256
      t:BinaryExchange [ ValueType= http://schemas.xmlsoap.org/ws/2005/02/trust/tlsnego EncodingType=http://docs.oasis-open.org/wss/2004/01/oa
        FgMBAEEBAAA9AwFMGmR0pPsTwgcXqSxflRjMXs0DZdYMTohx5lJ9HDPiAgAAFgAEAAUACgAJAGQAYgADAAYAEwASAGMBAA==
```

Figure 36: Capture 2 showing binary security token

Figure 36 shows the SOAP envelope, the header and the body of the message. The header section explains that the 'Action' is invoked from client which follows (*"/RST/ISSUE" specification*) in order to issue a token for communication, and each action is assigned a message number. The body of the message explains that when the client enters its username/ password, the WS-Security requests for a '*security token'* for the credentials provided by the client. This token is used for further communication with the service. The '*token type'* specifies the type of token requested and '*request type'* specifies that the request is generated in order to establish the trust for the client request to the service. Last line '*Binary exchange'* describes the binary security token that is generated from the client credentials that is used for communication with the service *("FghBAEEAAA….found in the last line" is the encrypted token generated for binary exchange*).

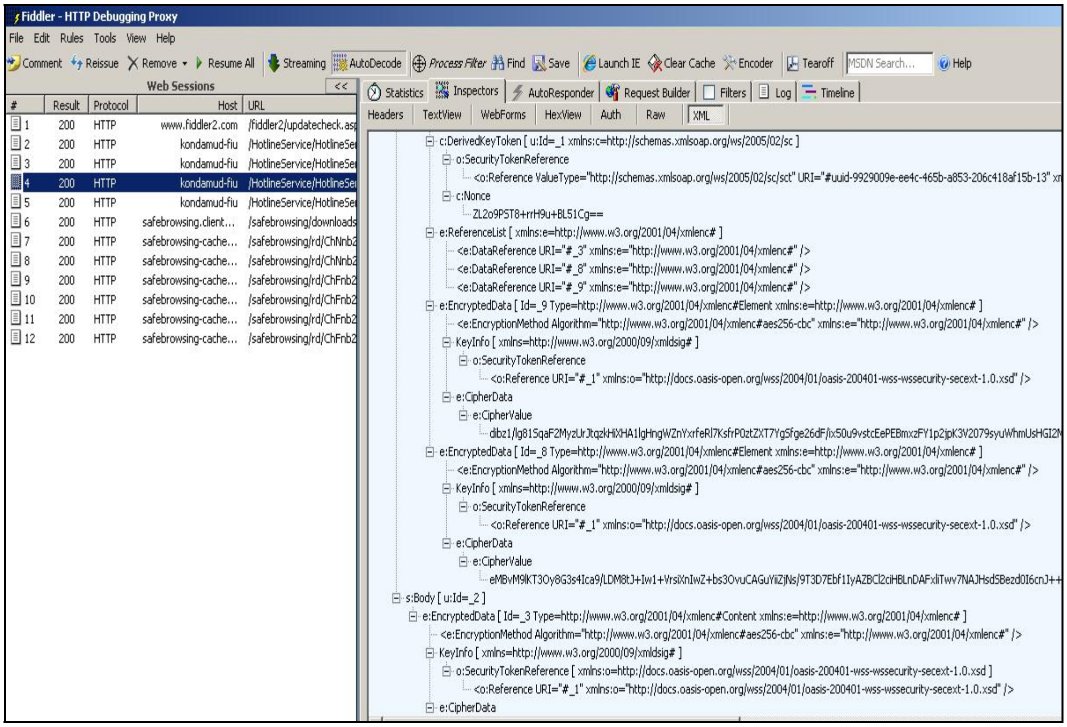

Fiddler - HTTP Debugging Proxy
File Edit Rules Tools View Help
Comment Reissue Remove Resume All Streaming AutoDecode Process Filter Find Save Launch IE Clear Cache Encoder Tearoff MSDN Search... Help
Web Sessions
<<
Statistics Inspectors AutoResponder Request Builder Filters Log Timeline
Headers TextView WebForms HexView Auth Raw XML

#	Result	Protocol	Host	URL
1	200	HTTP	www.fiddler2.com	/fiddler2/updatecheck.asp
2	200	HTTP	kondamud-fiu	/HotlineService/HotlineSer
3	200	HTTP	kondamud-fiu	/HotlineService/HotlineSer
4	200	HTTP	kondamud-fiu	/HotlineService/HotlineSer
5	200	HTTP	kondamud-fiu	/HotlineService/HotlineSer
6	200	HTTP	safebrowsing.client...	/safebrowsing/downloads
7	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChNnb2
8	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChNnb2
9	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
10	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
11	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
12	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2

```
⊟ s:Envelope [ xmlns:s=http://www.w3.org/2003/05/soap-envelope xmlns:a=http://www.w3.org/2005/08/addressing ]
  ⊟ s:Header
    ⊟ a:Action [ s:mustUnderstand=1 ]
        http://schemas.xmlsoap.org/ws/2005/02/trust/RSTR/Issue
    ⊟ a:MessageID
        urn:uuid:421a18c0-dc61-4672-a4a7-bd12ed084c00
    ⊟ a:ReplyTo
      ⊟ a:Address
          http://www.w3.org/2005/08/addressing/anonymous
    ⊟ a:To [ s:mustUnderstand=1 ]
        http://kondamud-fiu/HotlineService/HotlineService.svc
  ⊟ s:Body
    ⊟ t:RequestSecurityTokenResponse [ Context=uuid-d7f70804-f176-4632-9d68-e6ddeea55b40-1 xmlns:t=http://schemas.xmlsoap.org/ws/2005/02/trust
      ⊟ t:BinaryExchange [ ValueType= http://schemas.xmlsoap.org/ws/2005/02/trust/tlsnego EncodingType=http://docs.oasis-open.org/wss/2004/01/oas
          FgMBAIYQAACCAICe2GYCQtoOvmvqfet4bPIMqo5QXwItwEGDjwim9YbcF3NqhiUIJEfHnrwwhIZSSAv+PNxWnxYACecYR78f3y+d7+WMUg7xbwH
```

Figure 37: Binary security token for exchange

The figure 35 shows that the token (“FgMEA..”) generated from the previous figure is accepted to be used for Binary exchange. This can be seen from the ‘*Request security token response’* section. (*The token can be found from the last line of the fiddler display.)* It can also be observed that this is same token generated in the figure 35. This explains that the token issued by WS-Security specification is accepted for further communication with the service.

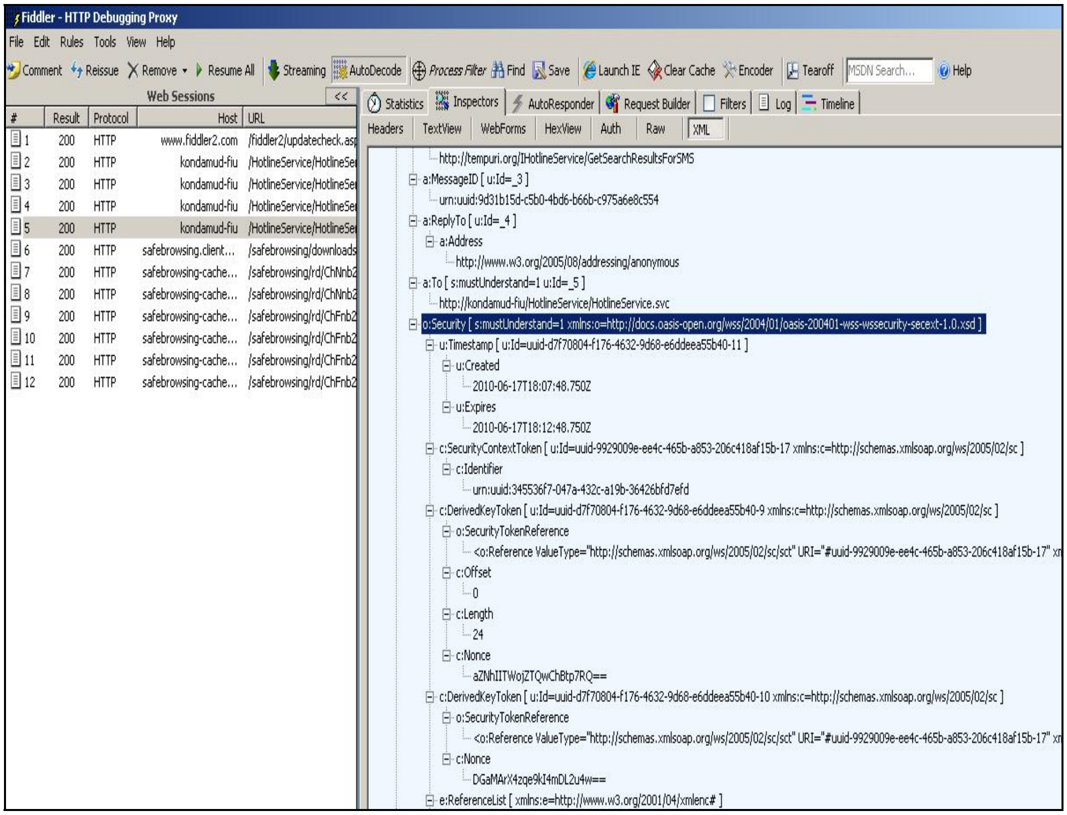

Fiddler - HTTP Debugging Proxy
File Edit Rules Tools View Help
Comment Reissue Remove Resume All Streaming AutoDecode Process Filter Find Save Launch IE Clear Cache Encoder Tearoff MSDN Search... Help
Web Sessions
Statistics Inspectors AutoResponder Request Builder Filters Log Timeline
Headers TextView WebForms HexView Auth Raw XML

#	Result	Protocol	Host	URL
1	200	HTTP	www.fiddler2.com	/fiddler2/updatecheck.asp
2	200	HTTP	kondamud-fiu	/HotlineService/HotlineSe
3	200	HTTP	kondamud-fiu	/HotlineService/HotlineSe
4	200	HTTP	kondamud-fiu	/HotlineService/HotlineSe
5	200	HTTP	kondamud-fiu	/HotlineService/HotlineSe
6	200	HTTP	safebrowsing.client...	/safebrowsing/downloads
7	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChNnb2
8	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChNnb2
9	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
10	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
11	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
12	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2

```
    c:DerivedKeyToken [ u:Id=_1 xmlns:c=http://schemas.xmlsoap.org/ws/2005/02/sc ]
      o:SecurityTokenReference
        <o:Reference ValueType="http://schemas.xmlsoap.org/ws/2005/02/sc/sct" URI="#uuid-9929009e-ee4c-465b-a853-206c418af15b-13" xr
      c:Nonce
        ZL2o9PST8+rrH9u+BL51Cg==
    e:ReferenceList [ xmlns:e=http://www.w3.org/2001/04/xmlenc# ]
      <e:DataReference URI="#_3" xmlns:e="http://www.w3.org/2001/04/xmlenc#" />
      <e:DataReference URI="#_8" xmlns:e="http://www.w3.org/2001/04/xmlenc#" />
      <e:DataReference URI="#_9" xmlns:e="http://www.w3.org/2001/04/xmlenc#" />
    e:EncryptedData [ Id=_9 Type=http://www.w3.org/2001/04/xmlenc#Element xmlns:e=http://www.w3.org/2001/04/xmlenc# ]
      <e:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" xmlns:e="http://www.w3.org/2001/04/xmlenc#" />
      KeyInfo [ xmlns=http://www.w3.org/2000/09/xmldsig# ]
        o:SecurityTokenReference
          <o:Reference URI="#_1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" />
      e:CipherData
        e:CipherValue
          dibz1/lg81SqaF2MyzUrJtqzkHiXHA1lgHngWZnYxrfeRl7KsfrP0ztZXT7YgSfge26dF/ix50u9vstcEePEBmxzFY1p2jpK3V2079syuWhmUsHGI2M
    e:EncryptedData [ Id=_8 Type=http://www.w3.org/2001/04/xmlenc#Element xmlns:e=http://www.w3.org/2001/04/xmlenc# ]
      <e:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" xmlns:e="http://www.w3.org/2001/04/xmlenc#" />
      KeyInfo [ xmlns=http://www.w3.org/2000/09/xmldsig# ]
        o:SecurityTokenReference
          <o:Reference URI="#_1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" />
      e:CipherData
        e:CipherValue
          eMBvM9lKT3Oy8G3s4Ica9/LDM8tJ+Iw1+VrsiXnIwZ+bs3OvuCAGuYiiZjNs/9T3D7Ebf1IyAZBCl2ciHBLnDAFxliTwv7NAJHsdSBezd0I6cnJ++
  s:Body [ u:Id=_2 ]
    e:EncryptedData [ Id=_3 Type=http://www.w3.org/2001/04/xmlenc#Content xmlns:e=http://www.w3.org/2001/04/xmlenc# ]
      <e:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" xmlns:e="http://www.w3.org/2001/04/xmlenc#" />
      KeyInfo [ xmlns=http://www.w3.org/2000/09/xmldsig# ]
        o:SecurityTokenReference [ xmlns:o=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd ]
          <o:Reference URI="#_1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" />
      e:CipherData
```

Figure 38: Binary security token in header

Figure 38 shows that the '*binary security token'* generated, is being sent in the message header to the service for further communication. But for security remains the security token should be encrypted. The first line shows that a Derived *'key token'* will be generated from the previous token reference. The reference list explains the data reference taken from the previous figures. '*EncryptedData'* explains the details about the encryption algorithm being used and the specific security token for which the encrypted data is to be generated. '*CipherData'* explains the cipher text that is generated after communication. The body section contains summary of the information generated from the header, and contains references to the WS-Security protocol to verify that the tokens generated from communication are following the WS-Security standard.


Fiddler - HTTP Debugging Proxy
File Edit Rules Tools View Help
Comment Reissue Remove Resume All Streaming AutoDecode Process Filter Find Save Launch IE Clear Cache Encoder Tearoff MSDN Search... Help
Web Sessions
Statistics Inspectors AutoResponder Request Builder Filters Log Timeline
Headers TextView WebForms HexView Auth Raw XML

#	Result	Protocol	Host	URL
1	200	HTTP	www.fiddler2.com	/fiddler2/updatecheck.asp
2	200	HTTP	kondamud-fiu	/HotlineService/HotlineSe
3	200	HTTP	kondamud-fiu	/HotlineService/HotlineSe
4	200	HTTP	kondamud-fiu	/HotlineService/HotlineSe
5	200	HTTP	kondamud-fiu	/HotlineService/HotlineSe
6	200	HTTP	safebrowsing.client...	/safebrowsing/downloads
7	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChNnb2
8	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChNnb2
9	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
10	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
11	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
12	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2

```
    http://tempuri.org/IHotlineService/GetSearchResultsForSMS
a:MessageID [ u:Id=_3 ]
    urn:uuid:9d31b15d-c5b0-4bd6-b66b-c975a6e8c554
a:ReplyTo [ u:Id=_4 ]
  a:Address
      http://www.w3.org/2005/08/addressing/anonymous
a:To [ s:mustUnderstand=1 u:Id=_5 ]
    http://kondamud-fiu/HotlineService/HotlineService.svc
o:Security [ s:mustUnderstand=1 xmlns:o=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd ]
  u:Timestamp [ u:Id=uuid-d7f70804-f176-4632-9d68-e6ddeea55b40-11 ]
    u:Created
        2010-06-17T18:07:48.750Z
    u:Expires
        2010-06-17T18:12:48.750Z
  c:SecurityContextToken [ u:Id=uuid-9929009e-ee4c-465b-a853-206c418af15b-17 xmlns:c=http://schemas.xmlsoap.org/ws/2005/02/sc ]
    c:Identifier
        urn:uuid:345536f7-047a-432c-a19b-36426bfd7efd
  c:DerivedKeyToken [ u:Id=uuid-d7f70804-f176-4632-9d68-e6ddeea55b40-9 xmlns:c=http://schemas.xmlsoap.org/ws/2005/02/sc ]
    o:SecurityTokenReference
        <o:Reference ValueType="http://schemas.xmlsoap.org/ws/2005/02/sc/sct" URI="#uuid-9929009e-ee4c-465b-a853-206c418af15b-17" xn
    c:Offset
        0
    c:Length
        24
    c:Nonce
        aZNhIITWojZTQwChBtp7RQ==
  c:DerivedKeyToken [ u:Id=uuid-d7f70804-f176-4632-9d68-e6ddeea55b40-10 xmlns:c=http://schemas.xmlsoap.org/ws/2005/02/sc ]
    o:SecurityTokenReference
        <o:Reference ValueType="http://schemas.xmlsoap.org/ws/2005/02/sc/sct" URI="#uuid-9929009e-ee4c-465b-a853-206c418af15b-17" xn
    c:Nonce
        DGaMArX4zqe9kI4mDL2u4w==
  e:ReferenceList [ xmlns:e=http://www.w3.org/2001/04/xmlenc# ]
```

Figure 39: Binary security token inserted in header

Figure 39 shows the last message number: 5 generated from the service. '*Security'* tag which contains references to all the information, the algorithms, the cipher data generated from fig.38. *All these references found under the Security tag explains that the above details generated from client are accepted by the service, and are used to verify the security specifications of the client request*. The highlighted security tag also includes '*ws-securit'y* which explains that the communication is being secured under the specific protocol standards, and is protected from attacks.

Result : From the above discussion we observe that Oakridge which is one of the client application for KMIT , has successfully integrated the hotline service, and could get the results published on KMIT, into its own application.

10.2 Experimental result -2:

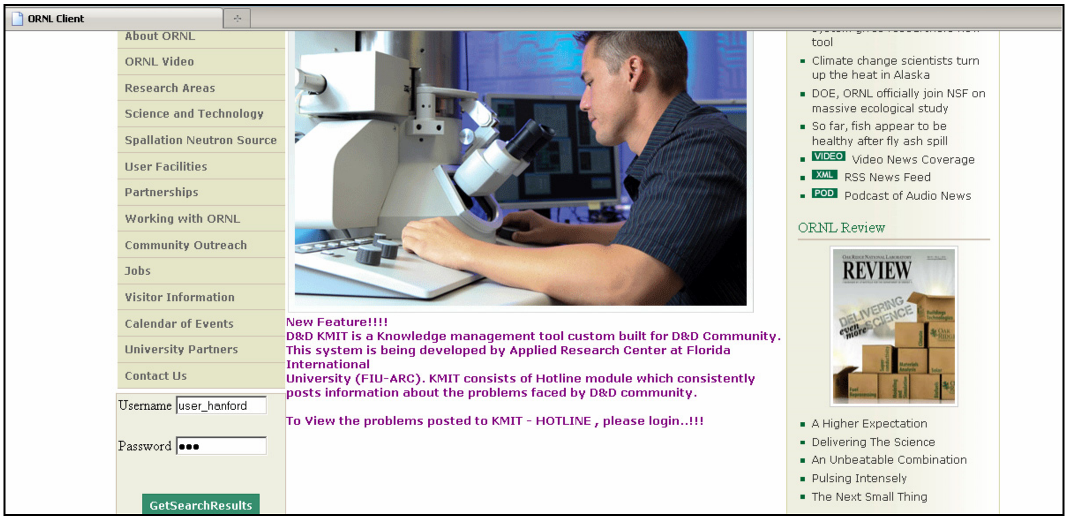
This is a fault test case to check how the Hotline Service security model ***does not authenticate*** *a client application with a wrong username and password.*

Credentials: username: user; password: pw

Service Certificate: tempCert signed by RootCA

Client: Oak Ridge National Laboratories application

Expected Output: A wrong username and password pair are not authenticated by the Service. Hence, it throws an error asking the application to try again with valid credentials.



Username user_hanford

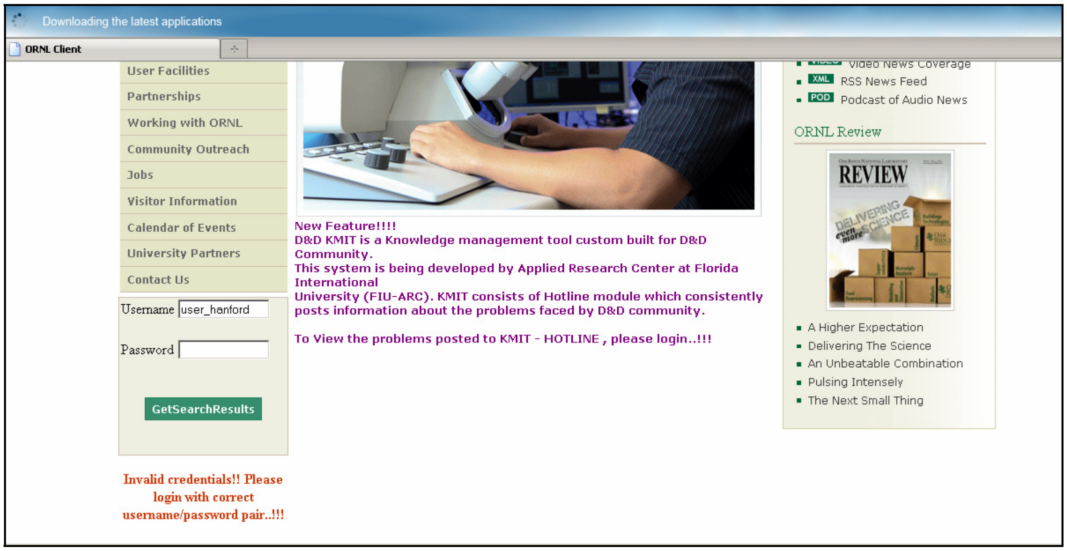
Password ●●●

New Feature!!!!
D&D KMIT is a Knowledge management tool custom built for D&D Community.
This system is being developed by Applied Research Center at Florida International
University (FIU-ARC). KMIT consists of Hotline module which consistently posts information about the problems faced by D&D community.

To View the problems posted to KMIT - HOTLINE , please login..!!!

Figure 40: Client application trying to access service with wrong password

Figure 40 displays the client application trying to access the service with invalid credentials.



Username user_hanford

Password

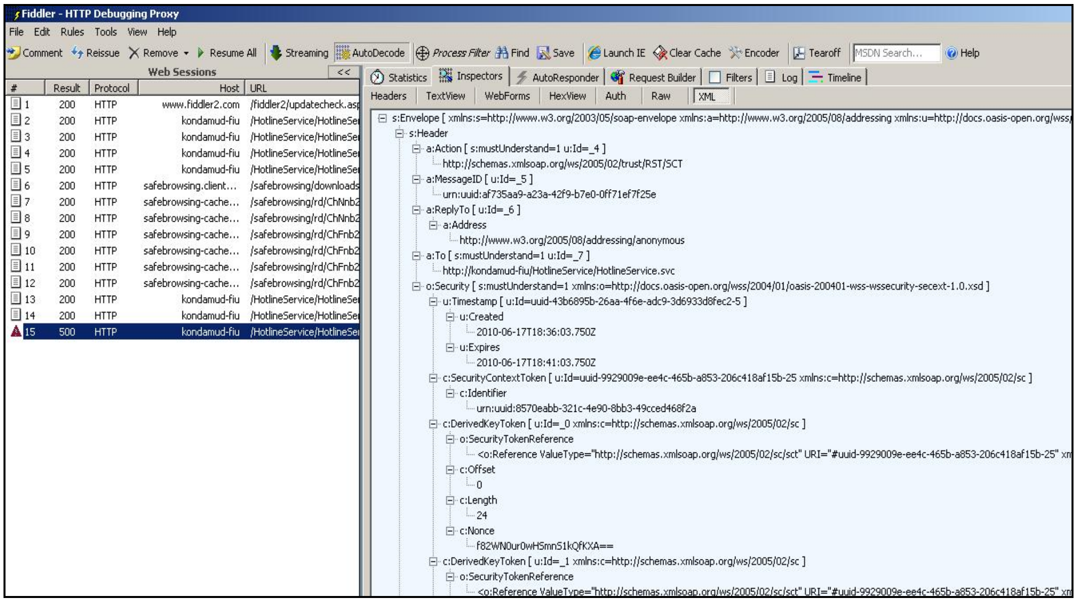
New Feature!!!!
D&D KMIT is a Knowledge management tool custom built for D&D Community.
This system is being developed by Applied Research Center at Florida International University (FIU-ARC). KMIT consists of Hotline module which consistently posts information about the problems faced by D&D community.

To View the problems posted to KMIT - HOTLINE , please login..!!!

Invalid credentials!! Please login with correct username/password pair..!!!

Figure 41: Client displaying error message

Figure 41 displays the client application throwing an error on entering the username and wrong password, which explains that the service could not validate the client credentials.


Fiddler - HTTP Debugging Proxy
File Edit Rules Tools View Help
Comment Reissue Remove Resume All Streaming AutoDecode Process Filter Find Save Launch IE Clear Cache Encoder Tearoff MSDN Search... Help
Web Sessions
<<
Statistics Inspectors AutoResponder Request Builder Filters Log Timeline
Headers TextView WebForms HexView Auth Raw XML

#	Result	Protocol	Host	URL
1	200	HTTP	www.fiddler2.com	/fiddler2/updatecheck.asp
2	200	HTTP	kondamud-fiu	/HotlineService/HotlineSer
3	200	HTTP	kondamud-fiu	/HotlineService/HotlineSer
4	200	HTTP	kondamud-fiu	/HotlineService/HotlineSer
5	200	HTTP	kondamud-fiu	/HotlineService/HotlineSer
6	200	HTTP	safebrowsing.client...	/safebrowsing/downloads
7	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChNnb2
8	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChNnb2
9	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
10	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
11	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
12	200	HTTP	safebrowsing-cache...	/safebrowsing/rd/ChFnb2
13	200	HTTP	kondamud-fiu	/HotlineService/HotlineSer
14	200	HTTP	kondamud-fiu	/HotlineService/HotlineSer
15	500	HTTP	kondamud-fiu	/HotlineService/HotlineSer

```
s:Envelope [ xmlns:s=http://www.w3.org/2003/05/soap-envelope xmlns:a=http://www.w3.org/2005/08/addressing xmlns:u=http://docs.oasis-open.org/wss/
  s:Header
    a:Action [ s:mustUnderstand=1 u:Id=_4 ]
      http://schemas.xmlsoap.org/ws/2005/02/trust/RST/SCT
    a:MessageID [ u:Id=_5 ]
      urn:uuid:af735aa9-a23a-42f9-b7e0-0ff71ef7f25e
    a:ReplyTo [ u:Id=_6 ]
      a:Address
        http://www.w3.org/2005/08/addressing/anonymous
    a:To [ s:mustUnderstand=1 u:Id=_7 ]
      http://kondamud-fiu/HotlineService/HotlineService.svc
    o:Security [ s:mustUnderstand=1 xmlns:o=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd ]
      u:Timestamp [ u:Id=uuid-43b6895b-26aa-4f6e-adc9-3d6933d8fec2-5 ]
        u:Created
          2010-06-17T18:36:03.750Z
        u:Expires
          2010-06-17T18:41:03.750Z
      c:SecurityContextToken [ u:Id=uuid-9929009e-ee4c-465b-a853-206c418af15b-25 xmlns:c=http://schemas.xmlsoap.org/ws/2005/02/sc ]
        c:Identifier
          urn:uuid:8570eabb-321c-4e90-8bb3-49cced468f2a
      c:DerivedKeyToken [ u:Id=_0 xmlns:c=http://schemas.xmlsoap.org/ws/2005/02/sc ]
        o:SecurityTokenReference
          <o:Reference ValueType="http://schemas.xmlsoap.org/ws/2005/02/sc/sct" URI="#uuid-9929009e-ee4c-465b-a853-206c418af15b-25" xm
        c:Offset
          0
        c:Length
          24
        c:Nonce
          f82WN0ur0wHSmnS1kQfKXA==
      c:DerivedKeyToken [ u:Id=_1 xmlns:c=http://schemas.xmlsoap.org/ws/2005/02/sc ]
        o:SecurityTokenReference
          <o:Reference ValueType="http://schemas.xmlsoap.org/ws/2005/02/sc/sct" URI="#uuid-9929009e-ee4c-465b-a853-206c418af15b-25" xm
```

Figure 42: Error showing authentication failure

Background process/ Discussion: Figure 42 explains that an error has occurred at request number 15 and the fiddler has stopped generating further requests. It can be understood from fiddler information that the binary security token has been generated from request numbers 13 and 14 in order to communicate with the service, but has failed to get validated from the service end by the error displayed from the service. The raw displays it the binary security token that was sent as part of the message which was not validated and so the service has not responded to the client request to sending the required information.

Result: Test has been passed and the expected output is displayed.

10.3 Experimental Result – 3:

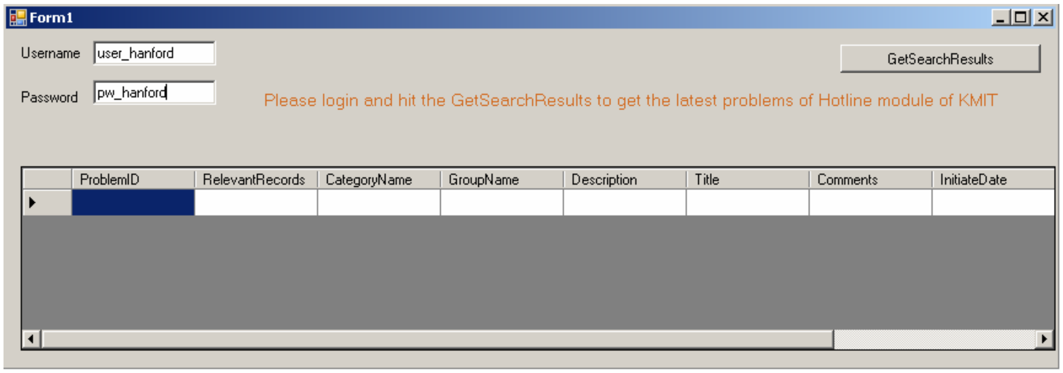
This is a fault test case that is performed from the service end. The service is given reference to the certificate which is not genuine and which does not contain the corresponding private key pair for the public key generated by Root CA. This fails the authentication model and throws an error. The data is not displayed on the client application.

Client Credentials: username: user_hanford ; password : pw_hanford

Service Certificate: test (created for test purposes not signed by RootCA)

Client: Forms application

Expected output: According to the hypothesis, on entering the username and password and clicking the button 'GetSearchResults', the client should talk to the Web Service and display the top five problems of KMIT in its own application. But as the service is referenced with wrong certificate, it is expected not to validate the credentials from the client. Hence, an error should be thrown.



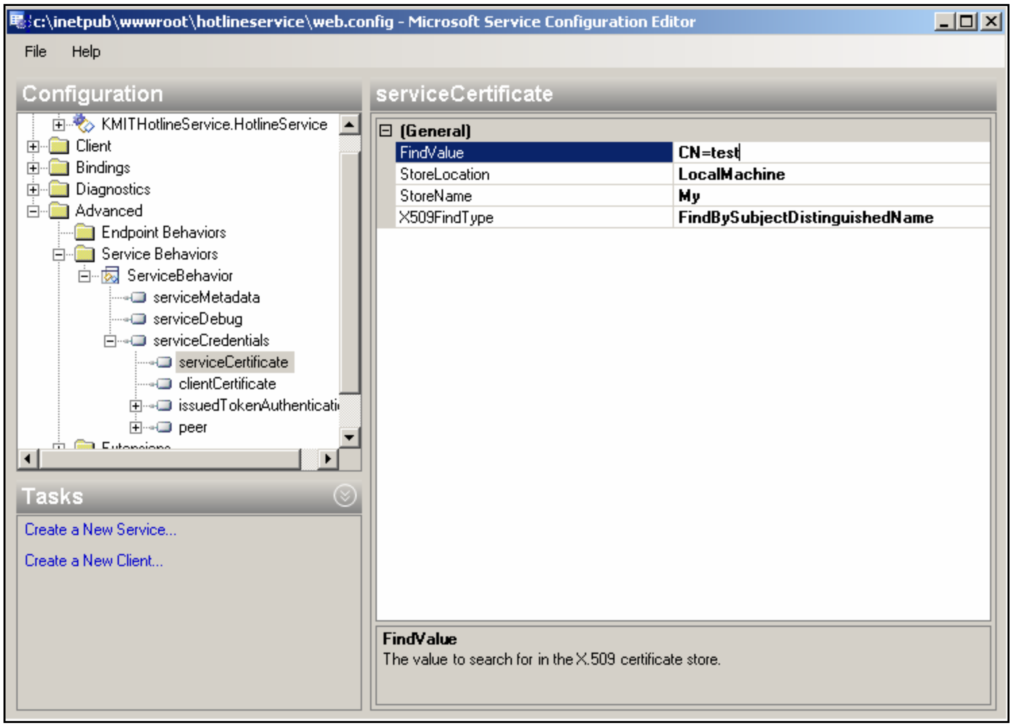
Username user_hanford

Password pw_hanford

Please login and hit the GetSearchResults to get the latest problems of Hotline module of KMIT

ProblemID	RelevantRecords	CategoryName	GroupName	Description	Title	Comments	InitiateDate

Figure 43: Client application with valid credentials



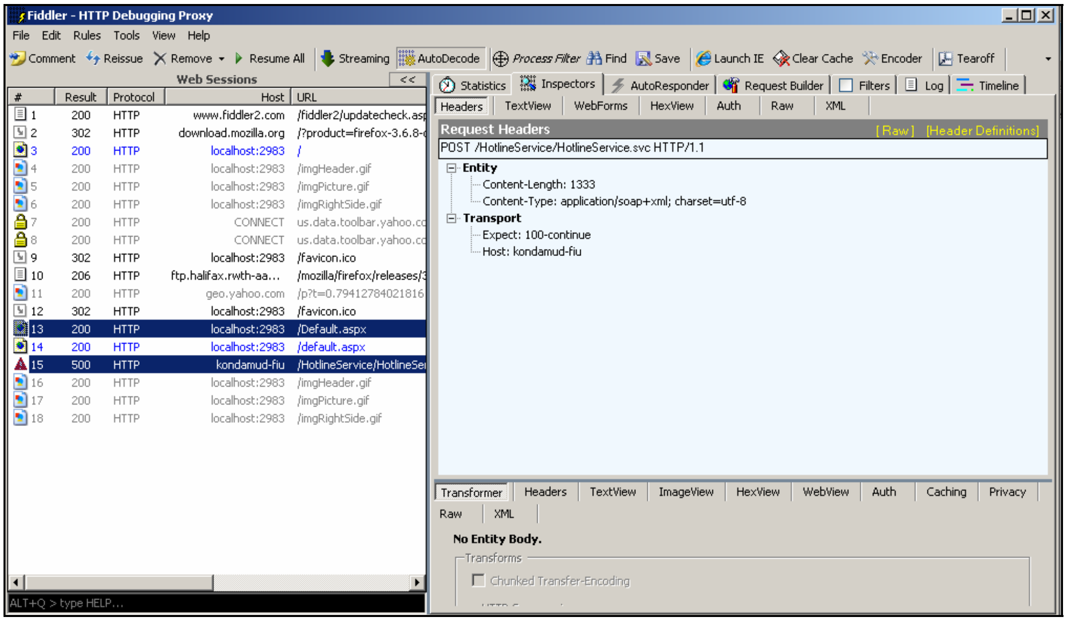
FindValue	CN=test
StoreLocation	LocalMachine
StoreName	My
X509FindType	FindBySubjectDistinguishedName

FindValue
The value to search for in the X.509 certificate store.

Figure 44: Service is given wrong reference of certificate

Figure 43 is the screenshot of the client application passing valid credentials in order to access the data from the service. In order to explain the interoperability where the client can be any application, we are using a windows forms application to access service. The password textbox is made visible in order to show that the same password is being used to generate the result. Figure 44 displays the screenshot where the service is given reference to a wrong certificate other than ‘tempCert’ signed by root CA.

Background process/Discussion: When the service is referenced with a certificate not signed by RootCA, it can be observed that fiddler generates an error message which explains that the authentication mechanism failed to validate the client credentials.


Fiddler - HTTP Debugging Proxy
File Edit Rules Tools View Help
Comment Reissue Remove Resume All Streaming AutoDecode Process Filter Find Save Launch IE Clear Cache Encoder Tearoff
Web Sessions
Statistics Inspectors AutoResponder Request Builder Filters Log Timeline
Headers TextView WebForms HexView Auth Raw XML
Transformer Headers TextView ImageView HexView WebView Auth Caching Privacy
Raw XML
ALT+Q > type HELP...

#	Result	Protocol	Host	URL
1	200	HTTP	www.fiddler2.com	/fiddler2/updatecheck.asp
2	302	HTTP	download.mozilla.org	/?product=firefox-3.6.8-c
3	200	HTTP	localhost:2983	/
4	200	HTTP	localhost:2983	/imgHeader.gif
5	200	HTTP	localhost:2983	/imgPicture.gif
6	200	HTTP	localhost:2983	/imgRightSide.gif
7	200	HTTP	CONNECT	us.data.toolbar.yahoo.co
8	200	HTTP	CONNECT	us.data.toolbar.yahoo.co
9	302	HTTP	localhost:2983	/favicon.ico
10	206	HTTP	ftp.halifax.rwth-aa...	/mozilla/firefox/releases/3
11	200	HTTP	geo.yahoo.com	/p?t=0.79412784021816
12	302	HTTP	localhost:2983	/favicon.ico
13	200	HTTP	localhost:2983	/Default.aspx
14	200	HTTP	localhost:2983	/default.aspx
15	500	HTTP	kondamud-fiu	/HotlineService/HotlineSer
16	200	HTTP	localhost:2983	/imgHeader.gif
17	200	HTTP	localhost:2983	/imgPicture.gif
18	200	HTTP	localhost:2983	/imgRightSide.gif

```
Request Headers                                   [Raw]  [Header Definitions]
POST /HotlineService/HotlineService.svc HTTP/1.1
Entity
    Content-Length: 1333
    Content-Type: application/soap+xml; charset=utf-8
Transport
    Expect: 100-continue
    Host: kondamud-fiu
```

No Entity Body.

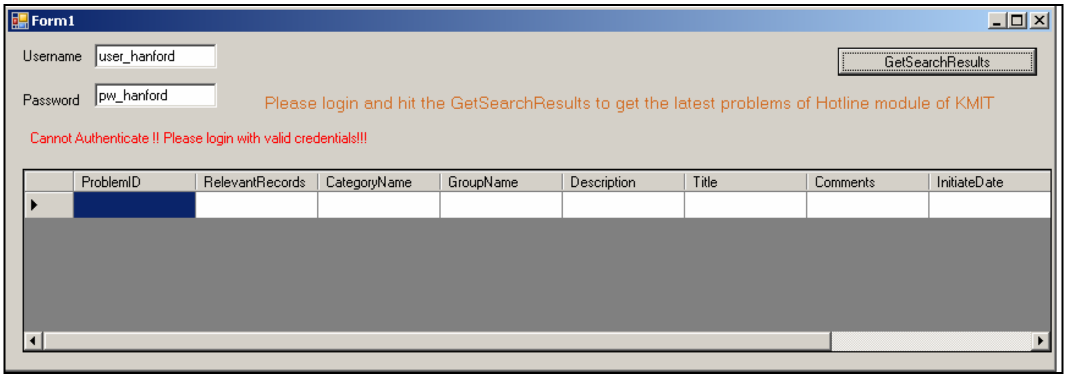
Transforms

□ Chunked Transfer-Encoding

Figure 45: Fiddler throwing an error at message 15 where the certificate could not decrypt the token with its private key

The fiddler generates a series of messages and the actual communication starts at message 13, where the local host is generated (client application). Message 13 and 14 allow for token generation and the encrypted form of token being sent to the service for further validation. But the error message is generated at message 15, where the service could not decrypt the encoded format of the token as the private key generated for that certificate is not being signed by RootCA, nor is the valid key pair for the public key.

This verifies that the service when referenced with a wrong certificate could not decrypt the client credentials and so, this proves the server side credential should also be referenced with its proper credential (certificate) in order to enable mutual authentication.



Username user_hanford

Password pw_hanford

Please login and hit the GetSearchResults to get the latest problems of Hotline module of KMIT

Cannot Authenticate !! Please login with valid credentials!!!

ProblemID	RelevantRecords	CategoryName	GroupName	Description	Title	Comments	InitiateDate

Figure 46: Client throwing an error saying that it is not authenticated.

Result: The service throws an exception when it is referenced with a wrong certificate. Hence security is maintained and test is passed.

11.0 CONCLUSION:

The Hotline Web service development and security address the "*secure authentication mechanism"* that satisfies the most important aspect of the security of the Hotline project. The study describes how the biggest problem- **'security'** of information exchanged through a Web Service is addressed. Results explained that WS-Security is the best security protocol for message security, while WCF is the best technology to help make web services secure. From the experiments it was shown that by creating a Web Service for the hotline module, features of KMIT could be integrated with its client application (ORNL) even though both of the applications are built using different technologies. Research about the various threat scenarios, security protocols, encryption standards and authentication models helped to design and develop a proper security model required to authenticate the client application for the hotline module of the knowledge management scenario.

FIDDLER is a remarkable de bugging tool which helped to debug the Hotline Web Service security implementation. The Hotline Service is de bugged using one best case and two fault test cases and the results are documented. The results show how good the fiddler can intercept the information exchanged, to provide the result. All the test cases are passed, proving the authentication of client applications accessing the Web Service. This study has paved a way for additional research about the other security features confidentiality and integrity which is recommended for future work to help develop the Hotline Service as a complete secured Web Service.

REFERENCES

[1] Rehman Mamoodi, Web Services background and Implementation, 18 August 2005
http://www.codeproject.com/KB/webservices/WebServices.aspx

[2] Hosting Services, MSDN Microsoft library,
http://msdn.microsoft.com/en-us/library/ms730158.aspx

[3] Jeffery C Broberg, Glossary for the OASIS Web Service interactive applications,
http://www.oasis-open.org/committees/wsia/glossary/wsia-draft-glossary-03.htm

[4] Understanding SOAP, MSDN Microsoft library
http://msdn.microsoft.com/en-us/library/ms995800.aspx

[5] UDDI, MSDN Microsoft library
http://msdn.microsoft.com/en-us/library/aa286530.aspx

[6] Erik Cristensen, Frnasicso Curbera, Greg Meredith, Sanjeeva Weerawarana, Web Services Description Language (WSDL) 1.1, 2001,
http://www.w3.org/TR/wsdl

[7] Dan Simon, The COM / DCOM Glossary, 1991 – 2001.
http://www.innovatia.com/software/papers/com.htm

[8] Knowledge Management Information Tool- KMIT – www.dndkm.org

[9] David Sprott and Lawrence Wilkes, Understanding Service Oriented Architecture, MSDN Microsoft Library, January 2004.
http://msdn.microsoft.com/en-us/library/aa480021.aspx

[10] S. C. Kendall, J. Waldo, A. Wollrath, G. Wyant, *A Note on Distributed Computing*, November 1994,
http://research.sun.com/techrep/1994/abstract-29.html

[11] Sandeep Kachru, Edward F. Gehringer, The Relative Advantages of Teaching Web Services in J2EE vs. .NET,
http://research.csc.ncsu.edu/efg/oo/papers/J2EE_.NET.pdf

[12] What is Windows Communication Foundation? , MSDN Microsoft library,
http://msdn.microsoft.com/en-us/library/ms731082 (v=VS.90).aspx

[13] Aaron Skonnard, Serialization in Windows Communication Foundation, August 2006, http://msdn.microsoft.com/en-us/magazine/cc163569.aspx

[14] Chris Peiris and Dennis Mulder , Hosting and Consuming Web Services– MSDN Microsoft, March 2007 - Revised May 2007,
http://msdn.microsoft.com/en-us/library/bb332338.aspx

[15]Security Overview, MSDN Microsoft library
http://msdn.microsoft.com/en-us/library/ms735093(v=VS.90).aspx

[16] Bob Atkinson, Giovanni Della-Libera, Satoshi Hada, Maryann Hondo, Phillip Hallam-Baker, Chris Kaler (Editor), Johannes Klein, Brian LaMacchia, Paul Leach, John Manferdelli, Hiroshi Maruyama, Anthony Nadalin, Nataraj Nagaratnam, Hemma Prafullchandra, John Shewchuk, Dan Simon, Web Services Security (WS-Security) version 1.0, April 5,2002.
http://schemas.xmlsoap.org/specs/ws-security/ws-security.htm

[17] Giovanni Della, Phillip Hallam- Baker, Maryann Hondo, Chris Kaler (Editor), Hiroshi Maruyama, Anthony Nadalin, Nataraj Nagaratnam, Hemma Prafullchandra, John Shewchuk, Kent Tamura, Hervey Wilson, Web Services Security Addendum, version 1.0, August 18, 2002.
http://xml.coverpages.org/WS-Security-Addendum200208.pdf

[18] Web Services Security: SOAP Message Security 1.1 (WS-Security 2004) OASIS Standard Specification, 1 February 2006,
http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[19] Security Architecture – MSDN Microsoft
http://msdn.microsoft.com/en-us/library/ms788756.aspx

[20] Use Username Authentication with the SQL Server Membership Provider and Message Security in WCF from Windows Forms – MSDN Microsoft
http://msdn.microsoft.com/en-us/library/cc949082.aspx

[21] OASIS Web Service Secure Exchange TC, OASIS WS-Trust 1.3 Oasis Standard, 19 March, 2007,
http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html

[22] OASIS Web Services Secure Exchange TC, OASIS WS-Secure Conversation 1.3 Oasis Standard, 1 March 2007,
http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.html

[23] OASIS Web Services Secure Exchange TC, WS-Security Policy 1.2 Oasis Standard, 1 July, 2007,
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.html

[24] Microsoft Corporation, Implementing Transport and Message Layer Security-MSDN Microsoft, December 2005,
http://msdn.microsoft.com/en-us/library/ff647370.aspx

[25] Message and Transport Security – MSDN Microsoft
http://msdn.microsoft.com/en-us/library/ff648863.aspx

[26] Create and Install Temporary Certificates in WCF for Message Security During Development, - MSDN Microsoft
http://msdn.microsoft.com/en-us/library/cc949011.aspx