Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 5, Issue. 10, October 2016, pg.141 – 160

Adapters for Service Version Compatibility

T.V.Ramana[1], Tesfalidet Tecle Fissehazion[2]
[1]Asst.Professor, Computer Engineering Department, Eritrea Institute of Technology, Asmara, Eritrea
[2]Head of the Computer Engineering Department, Eritrea Institute of Technology, Asmara, Eritrea
Venkataramana.t@gmail.com, tesstecle@gmail.com

Abstract: Web services are emerging as the next generation software systems in distributed computing. They are being widely adopted in implementing and integrating software components both within and across enterprises. In this paper, we consider the problem of interoperability among different web service technologies and web service versions.
The main objective of our research is to enable a client designed for an old version of a web service to successfully communicate with a newer version of the same service without making any changes on the client. To accomplish this objective, we propose the design of a service adapter – a web service that mediates the interactions among two services with different interfaces so that interoperability can occur. A key ingredient of the adaptation methodology is the use of mismatch patterns.
Accordingly, we have developed a WCF service adapter and the corresponding WCF and Java clients as part of our experiment and we have also tested our proposed methodology on a real world application.
In the second part of our experiment, we dealt with interoperability in two of the most widely used web service technologies (i.e. WCF and Metro). We experimented on BasicHttpBinding and WSHttpBinding, which are two of WCF's most commonly used protocols.

Key Words: service adapters, web services, WCF, Metro, SOA.

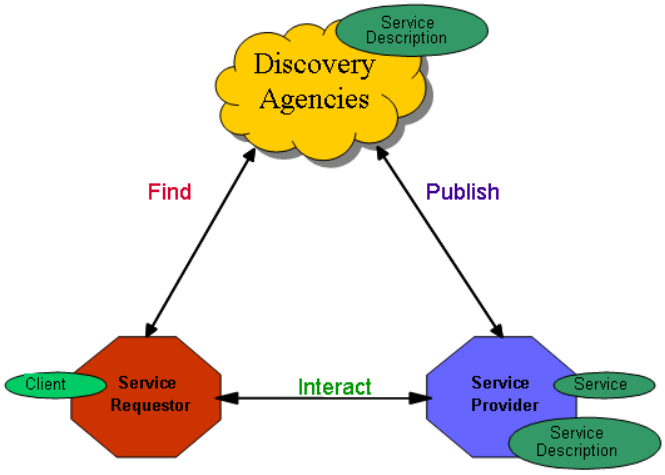
INTRODUCTION

Service-Oriented Architecture (SOA)

Definition: Is a computing paradigm that utilizes services as fundamental elements to support rapid, low-cost development of distributed applications in heterogeneous environments.

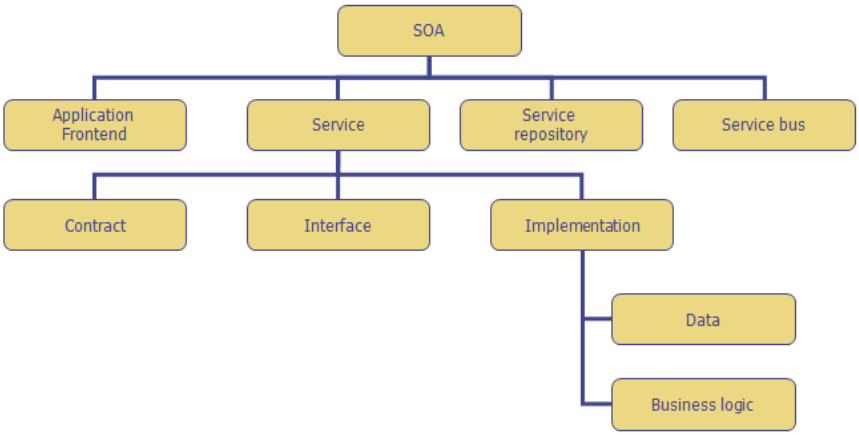
Common examples of service-oriented applications include sharing information, handling multistep processes such as reservation systems and online stores.

Three main parts of SOA



Elements of a SOA

SOA is a software architecture based on four key abstractions: application frontend, service, service repository and service bus.



Adapters

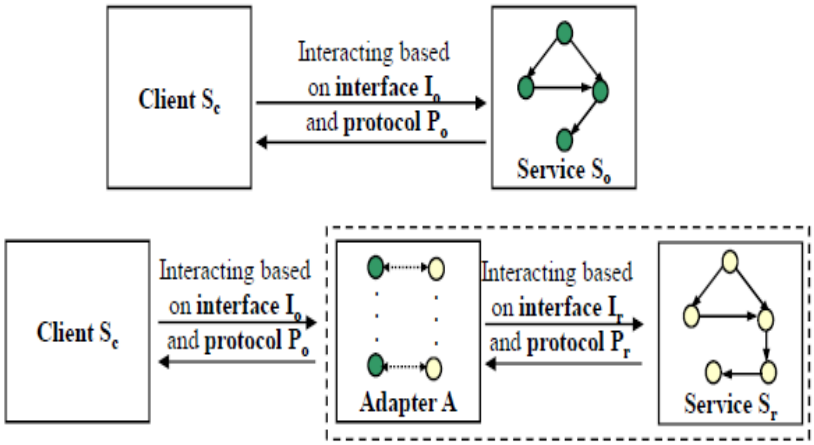
Adapters (wrappers) are design patterns that are used to convert the interface of a class into another interface clients expect.

The adapter translates calls to its interface into calls to the original interface, and the amount of code necessary to do this is typically small. The adapter is also responsible for transforming data into appropriate forms. For instance, if multiple boolean values are stored as a single integer (i.e. flags) but your consumer requires a 'true'/'false', the adapter would be responsible for extracting the appropriate values from the integer value.

There are two types of adapters: the *object adapter* which contains an instance of the class it wraps and the *class adapter* which uses multiple inheritance to achieve its goal.

Service adaptation refers to the process of generating a service (the adapter) that mediates the interactions among two services with different interfaces and protocols so that interoperability can occur. The need for adapters in Web services comes from two sources: one is the heterogeneity at the higher levels of the interoperability stack (e.g., at business-level interfaces and protocols), and the other is the high number and diversity of clients, each of which can support different interfaces and protocols, thereby generating the need for providing multiple faces to the same service.

An adapter allows achieving protocol replaceability



WCF METRO INTEROPERABILITY

Interoperability refers to the ability of different systems to work together seamlessly without any special effort. With an increase in service-oriented applications interoperability has become a major software engineering challenge.

WCF and Metro are two of the major web service development stacks available today.

Windows Communication Foundation

Windows Communication Foundation (WCF) is Microsoft's unified programming model for building service-oriented applications. It enables developers to build secure, reliable, transacted solutions that integrate across platforms and interoperate with existing investments.

WCF is designed in accordance with service oriented architecture principles to support distributed computing where services are consumed by consumers. Clients can consume multiple services and services can be consumed by multiple clients. Services are loosely coupled to each other. Services typically have a WSDL interface that any WCF client can use to consume the service, irrespective

of which platform the service is hosted on. WCF implements many advanced Web services (WS) standards such as WS-Addressing, WS-ReliableMessaging and WS-Security.

BasicHttpBinding and WSHttpBinding

WCF provides the BasicHttpBinding and the WSHttpBinding as two kinds of bindings that use Http as transport protocol. These are the most interoperable bindings that are useable.

Metro

Metro is an open source web services stack developed by Sun Microsystems. It is bundled with numerous applications servers such as GlassFish and Oracle WebLogic Server. Both Sun Microsystems and Microsoft have worked together to ensure interoperability between Metro and WCF.

WSIT

Web Services Interoperability Technology (WSIT) is an open-source project started by Sun Microsystems to develop the next-generation of web service technologies.

ADAPTERS FOR SERVICE VERSION COMPATIBILITY

Adapters, also known as wrappers, are not new to software design. They have been used in component based software development and to mediate between incompatible class interfaces and to transform data into appropriate forms.

In this section, we classify and describe mismatches that occur among different web services. Later we propose a methodology for developing adapters that can be used to mediate differences in web service versions; i.e. a client developed for service S_0 can successfully interact with service S_1 (a newer version of S_0) through the use of an adapter.

Mismatch Types and Solutions

Interoperability among web services requires that services use the same (or compatible) protocols, data formats, and semantics. To interact, services must have compatible:

- Interfaces (i.e., the set of operations supported by services),
- Business protocols (i.e., the allowed message exchange sequences).

A summary of protocol level mismatches are the following:

- *Ordering Mismatch*: This type of mismatch occurs when protocols P and Pr support the same messages but in different orders.
- *Extra Message Mismatch*: This mismatch occurs when protocol Pr sends a message that protocol P does not send.
- *Missing Message Mismatch*: This mismatch occurs when protocol Pr does not issue a message specified in the protocol P.
- *One-to-Many Message Mismatch*: This mismatch occurs when protocol P specifies a single message to achieve a functionality, while protocol Pr requires several messages for the same functionality.
- *Many-to-One Message Mismatch*: This mismatch occurs when protocol P specifies several messages to achieve a functionality, while protocol Pr requires only one message for the same functionality.

Interface Mismatch Types

Interface mismatch, also known as signature mismatch, occurs when the set of operations exposed by a service S are different from the set of operations exposed by service T.

Considering an old version S and new version T of a web service, the following interface incompatibilities can occur:

- *Missing methods*: If T removes or renames a method which was in the old service, S-clients may use this method and hence be incompatible with T.
- *Extra Methods*: If T adds a method that must be called by its clients to function properly. S clients fail to call this method and hence be incompatible with T.
- *Extra fields*: If T adds a field *f* to a method in the old service, S-clients do not use *f*, but T expects a value for *f* if it is mandatory. T may also add an extra field in method outputs.
- *Missing fields*: If T removes a field *f*, S-clients using this field are
- incompatible with T. Incompatibility occurs even if *f* is optional– if an S-
- client uses an optional field, it may be important to the application, and cannot be ignored.
- *Value-Space mismatches*: If S and T have different value-spaces for an input field *f*, where the value space for S.f is not a subset of the value space for T.f, values passed by S-clients may be disallowed for T resulting in incompatibility. For output fields, incompatibility occurs when value space of T.f is not a subset of value spaceofS.

Adapter Solutions to Interface Mismatches

As mentioned earlier, our solution to incompatibilities in service versions are adapters.

Table summarizes how an adapter resolves these mismatches.

Mismatch Type	Old Service S New Service T	Adapter Solution
Missing methods mismatch	• T removes a method • T renames a method	• Find a method that corresponds to the removed method • Use the new method name of T
Extra methods mismatch	• T adds a new method	• Get parameters required by new method from S or generate them • Call the new method
Extra fields mismatch	• T adds an extra input field to a method • T adds an extra output field	• Generate a value that does not affect the result • Drop the field
Missing fields mismatch	• T removes a field from input • T removes a field from output	• Store supplied fields and use when necessary • Try and generate the missing field value from the result
Value-Space mismatch	• Value space for input field of S not subset of value space for input field of T • Value space for output field of T is not a subset of value space for output field of S	• Supply a substitute value if possible • Send a constraint mismatch exception to the client

Table1 Summary of incompatibilities and corresponding adapter solutions

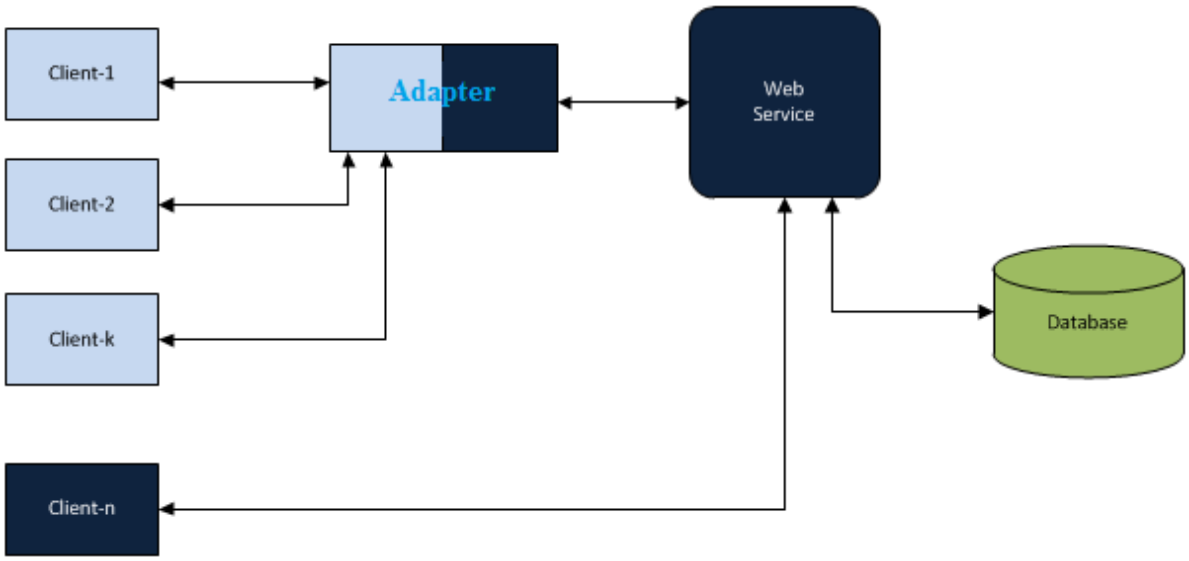
Architecture

The architecture is the blueprint for the system and therefore the implicit high-level plan for its construction.

Our system architecture consists of four main parts: the clients, the adapter, the web service and the database:

- *The clients*, written either in C# or Java, are the consumers of the services provided by the web service. They have a user interface to invoke the services. However, they do not include any kind of implementation in their design.

- *The adapter* acts as a mediator between old client versions and new web service versions. Internally, the adapter is both a client and a web service. However, the adapter does not implement the methods it exposes. Instead, it calls the methods exposed by the new version of the web service.
- *The web service* is the service provider that implements and exposes web methods. The web methods, with the assistance of the database, perform the required calculations and return the results. Additionally, the web service is responsible for checking the credentials provided by the client against the database before granting access to its web methods.
- *The database* resides in a database management system and stores all data. It also performs some calculations through the use of stored procedures. Illustrates the relationship among the different parts of the system.



General architecture of the system

Service Version 1 (S1)

This is the first and the simplest version of the WCF services. It uses BasicHttpBinding to configure and expose endpoints. S1 does not use any of the transport or message security modes. As a result, all data transmission between the client and the service are unencrypted.

Since BasicHttpBinding does not support sessions, this version of the WCF service is stateless. Figure shows the basic setup of S1 and its clients.

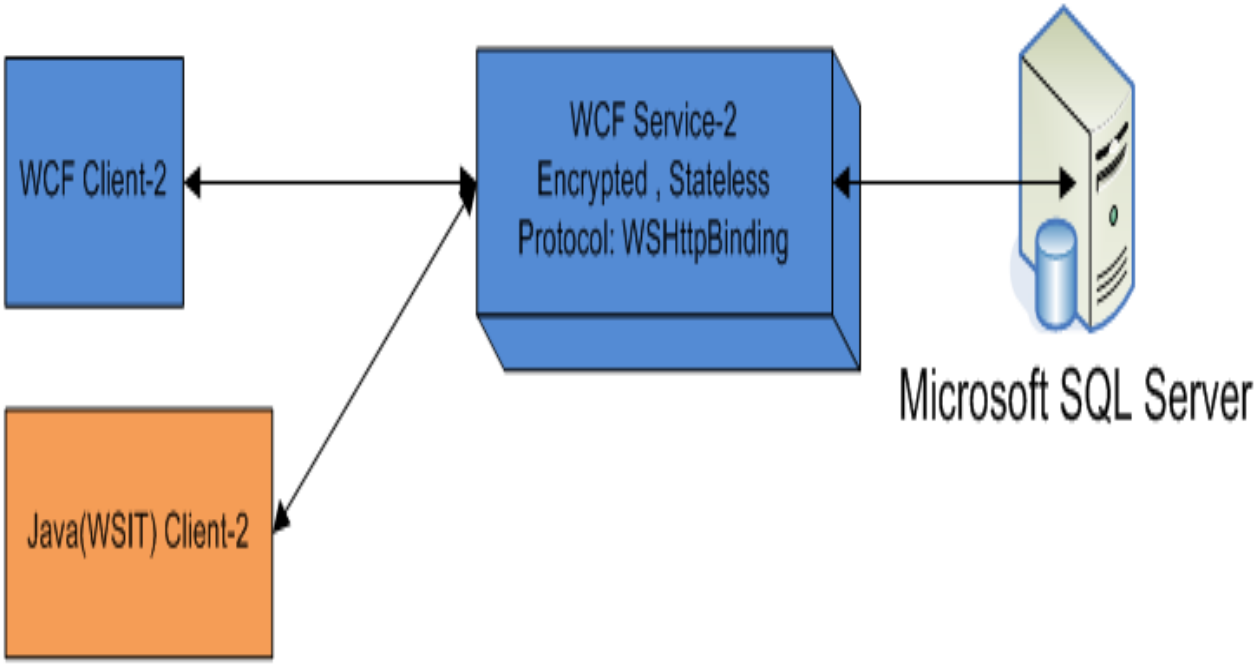


Figure basic setup for WCF client version 1, Java client version 1 and WCF service version 1

The web methods exposed by S1 are depicted in figure .

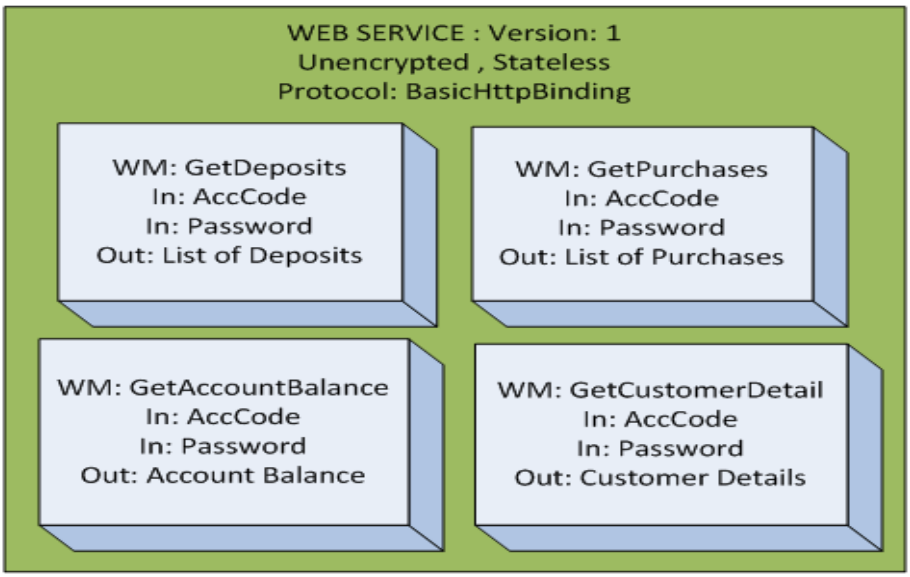


Figure the web methods exposed by S1

Table 2 gives a detailed description of the web methods exposed by S1.

Web Method	Input [data type]	Output [data type]	Function
GetDeposits	• Account code of the client [string] • Password of the client [string]	• List of deposits [array of Transactions]	Returns the list of deposits made by the client to date.
GetPurchases	• Account code of the client [string] • Password of the client [string]	• List of purchases [array of Transactions]	Returns the list of purchases made by the client to date.
GetAccountBalance	• Account code of the client [string] • Password of the client [string]	• The current balance of the supplied account code [decimal]	Sums all the deposits and all the purchases made to date and returns the difference.
GetCustomerDetail	• Account code of the client [string] • Password of the client [string]	• The details of the customer [customer]	Returns the details of the customer such as customer name, address etc…

Table2Details of the web methods exposed by s1

Service Version 2 (S2)

S2 is a more advanced version of S1. It uses WSHttpBinding as its binding protocol. By default, WSHttpBinding encrypts all message transmissions. Therefore, all communications between S2 and its clients are encrypted.

Figure basic setup for WCF client version 2, Java client version 2 and WCF service version 2

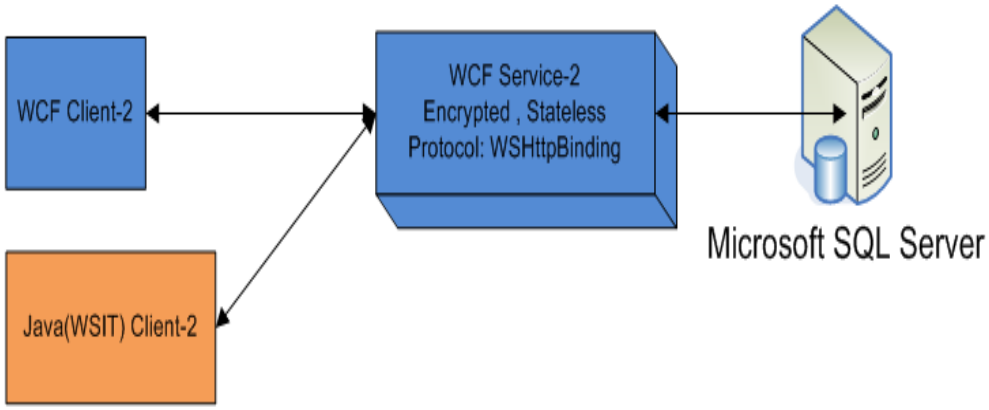
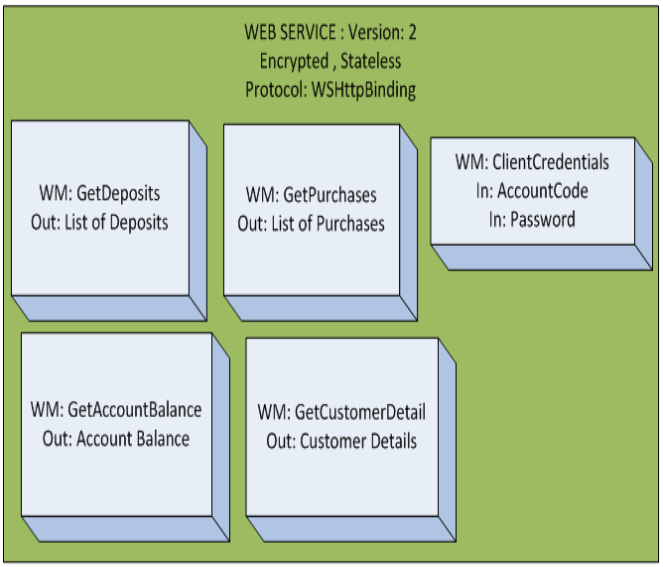


Figure the web methods exposed by S2



The service configuration of S2 is more complex than of S1. As in S1, it starts with the WSDL relative URL (i.e. MEX). And then it specifies how to find the X.509 certificate. The next part of the configuration (i.e. UserNameAuthentication) sets the validation mode and the custom function used for validation of username and password.

Unlike S1, S2 uses a message security scheme to encrypt all data transmission. To make the service compatible with Java clients, we set negotiateService Credential and establishSecurityContext

options to false. And since the default Java package does not support basic 256-bit encryption algorithm, we lowered it to basic 128-bit.

Apart from the binding protocol (i.e. wsHttpBinding), the rest of the configuration is similar to that of S1.

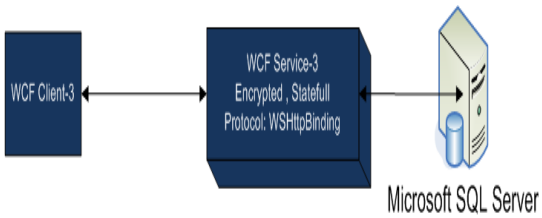
Service Version 3 (S3)

This version of the web service differs from S1 in two major areas. Firstly, the messages are encrypted and the web service is statefull - which means it can remember previous conversations with the client. Secondly, the methods it exposes are different from the methods S1 exposes as we will see shortly.

To make S3 a statefull web service, first we had to set the option establishSecurityContext to true thereby loosing interoperability with Java clients. We also had to set the service behavior InstanceContextMode to PerSession (as opposed to PerCall used in S1 and S2).

As a result of these changes, the implementation of the web method GetAccountBalance is a bit different from the previous two implementations. When a client calls this method for the first time, the web service does the actual calculations and returns the result to the client. However, it also stores the result in memory in case the client calls this method again. All subsequent calls to this web method return the result stored in memory and not from actual calculations. This can be a useful feature when database connections are expensive. This result will be active as long as the client does not close the session.

The below Figure shows the basic setup of S3 and its WCF client.



The web methods exposed by S3 are depicted in figure .

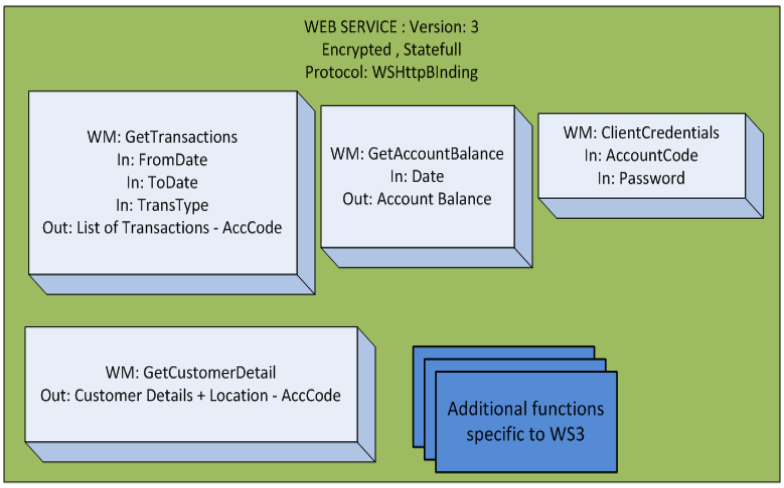


Figure the web methods exposed by S3

Figure the web methods exposed by S3

Web Method	Input	Output	Function
GetTransactions	• Start filter From Date [datetime] • End filter on To Date [datetime] • Transaction Type [Enumeration(D,P)]	• List of Transactions(deposits or purchases based on input TransType) without Account Code	Returns the list of deposits/purchases made by the client between the given dates.
GetAccountBalance	• Calculate until Date [datetime]	• The balance of the supplied account code at the specified date [decimal]	Sums all the deposits and all the purchases made until the specified date and returns the difference.
GetCustomerDetail		• The details of the customer plus the Location of the account [customer] without Account Code	Returns the details of the customer such as customer name, address, location etc…

Details of the web methods exposed by S3

Adapter Design

The most important part of our experiment was designing the adapter. The objective of our research was to enable a client designed for an old version of a web service to successfully communicate with a newer version of the same service without making any changes on the client. In our setup, this means enabling Client-1 communicate with S3 without making any changes to the client

Internally, the adapter has two components: the service component and the client component. The service component of the adapter exposes web methods identical to S1. This allows Client-1 to successfullycommunicate with the adapter. The similarities however end there. The adapter does not in fact implement the web methods. Instead it uses its client component to call methods exposed by S3; which implies the service S3 must be running for this setup to work properly.

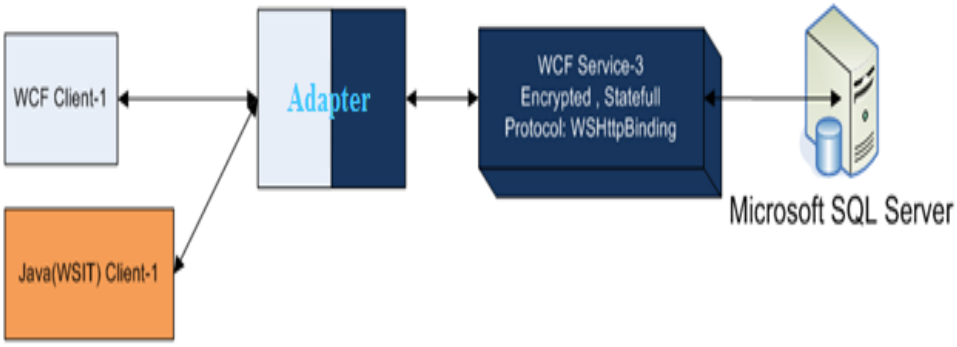


Figure basic setup for version 1 clients, WCF Ada pter and WCF service version 3

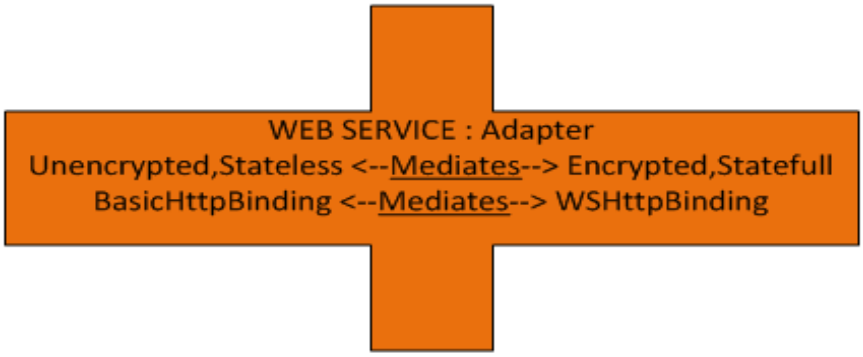
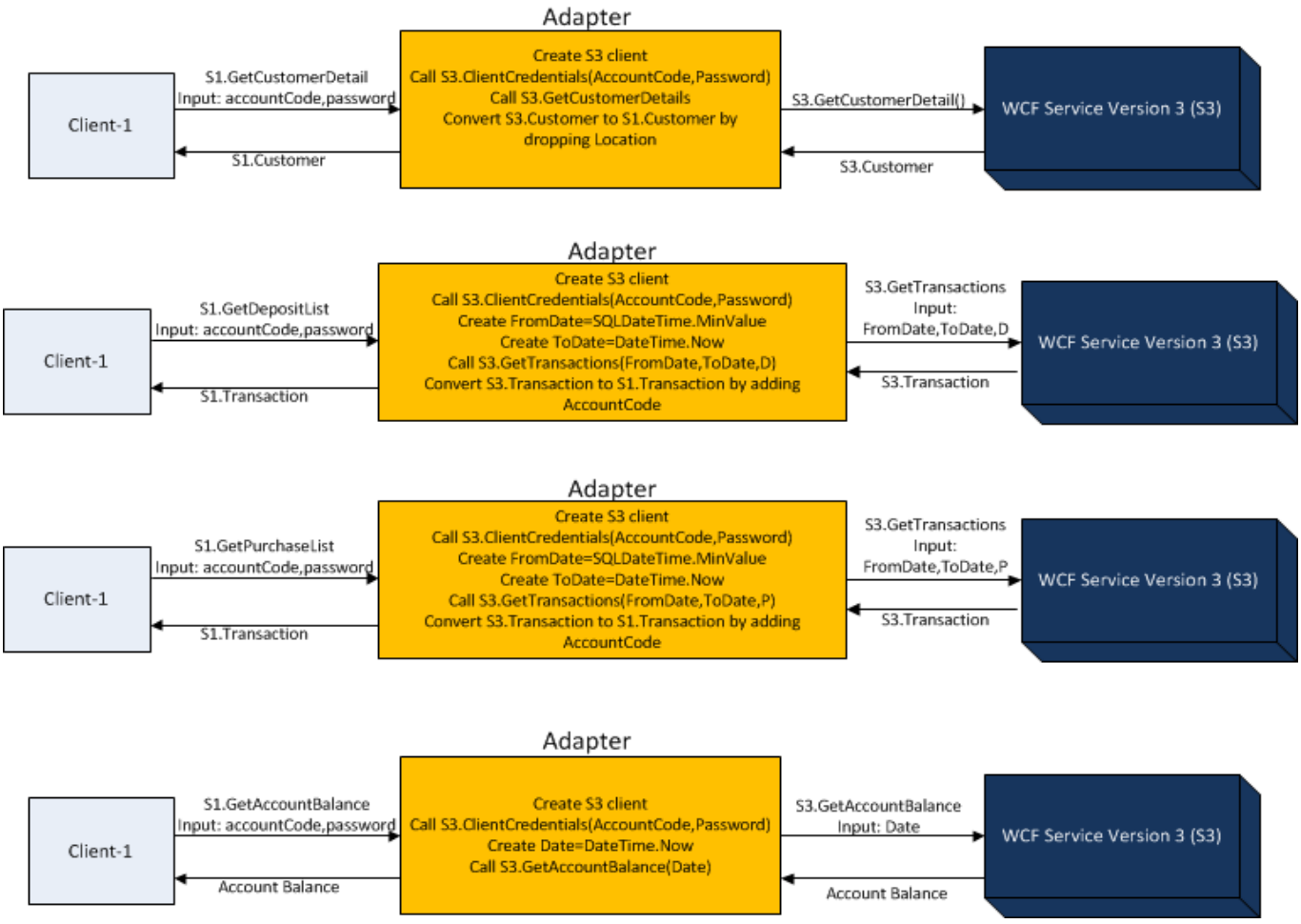


Figure adapter as a mediator between different protocols and data transmissions

The configuration file for the adapter contains both service configuration and client configuration allowing it to act as a service and a client. The service component uses BasicHttpBinding for clients

communicating with it. However, the client component of the adapter uses WSHttpBinding to connect with S3.



Internal design of the adapter

In the above we see the internal workings of the adapter. Client-1 has no knowledge that S1 has been replaced by the adapter since the service component of the adapter is identical to S1. When client-1 invokes the method GetCustomerDetail, it passes the customer's account code and password as input to the adapter. The adapter then creates a client that can communicate with service version 3 (S3). This client then calls the web methods ClientCredentials and GetCustomerDetails of S3 using the parameters passed to it by client-1. It will then convert the output of GetCustomerDetails of S3 (a customer object with an additional location field i.e. S3.cutomer) to a data type compatible with client-1 (i.e. S1.customer) by dropping the location field. Finally, the adapter forwards this data type to client-1 as output.

Results

Interoperability Results

One of the objectives of the research was to see the interoperability of different service-oriented application development technologies. For our experiments, we chose WCF (Windows Communication Foundation) from Microsoft and GlassFish Metro, the open-source web service stack.

Summary of experiment results for interoperability

Configuration	Interoperable	Analysis
Protocol: BasicHttpBinding *Security: None* *Data Transmission: Unencrypted, Stateless*	Yes	*BasicHttpBinding* represents a binding protocol to configure and expose endpoints that are able to communicate with clients and services that conform to the WS-I Basic Profile 1.1. Both WCF and GlassFish Metro are compatible with WS-I Basic Profile 1.1. By default, security in BasicHttpBinding is disabled. BasicHttpBinding does not support sessions
Protocol: WSHttpBinding *Security: Message, clientCredentialType="UserName", negotiateServiceCredential="false", algorithmSuite="Basic128", establishSecurityContext="false"* *Data Transmission: Encrypted, Stateless*	Yes	*WSHttpBinding* defines a secure, reliable, interoperable binding suitable for non-duplex service contracts. The binding implements the following specifications: WS-Reliable Messaging for reliability, and WS-Security for message security and authentication. The Default Algorithm Suite for Message Secuity is Basic256. However, Java requires a special library to support this suite.

Protocol: WSHttpBinding *Security: Message,* *clientCredentialType="UserName",* *negotiateServiceCredential="true",* *establishSecurityContext="true"* *Data Transmission: Encrypted, Statefull*	No	*negotiateServiceCredential*: Setting this property to true, requires WS-Trust and WS-SecureConversation to be supported by the client and the service. The protocols used (SPNego, TLSNego) to exchange tokens are not interoperable. *establishSecurityContext:* setting this option to true enables the parties to establish a secure session to reduce the overhead of one-off key exchange and validation. [31] Setting it to true requires that the remote party supports WS-SecureConversation.

Adapter Results

The next part of our experiment was to see if an adapter can be used to mediate the differences between S1 and S3 thereby enabling clients designed for S1 to successfully communicate with S3. we described the two types of clients S1 comprises: a WCF client and a Java client. In our experiments, by making use of an adapter we were able to make these clients call the web methods exposed by S3. Table summarizes the mismatch types resolved by our adapter.

We see that S3's getCustomerDetails method does not have the two input fields of S1's corresponding method causing a *missing input field mismatch*. We also notice that S3.Customer has an extra Location field which causes an *extra output field mismatch*. To authenticate its clients, S3 has to set the properties client.ClientCredentials.UserName.UserName and client.ClientCredentials.UserName.Password; which is the reason for *extra method mismatch*.

The web method getDepositList does not have a corresponding method in S3 causing *missing method mismatch*. On the other hand, getTransactions has extra fields which are not supplied by S1 clients which results in *extra input field mismatch*. The data type S3.Transaction does not include AccCode required by S1 clients causing *missing output field mismatch*.

The mismatches that occur due to getAccountBalance are similar to the previous cases.

Table Mismatches resolved by the adapter

S1 Web Methods	S3 Web Methods	Mismatches Resolved by the Adapter
getCustomerDetails Input: string accountCode Input: string password Output: S1.Customer S1.Customer string AccCode string Name string Address	getCustomerDetails Output: S3.Customer S3.Customer string Name string Address string Location client.ClientCredentials.UserName.UserName client.ClientCredentials.UserName.Password	• Missing input field • Extra output field • Extra method
getDepositList Input: string accountCode Input: string password Output: List<S1.Transaction> S1.Transaction string AccCode DateTime Date decimal Amount string Description	getTransactions Input: DateTime fromDate Input: DateTime toDate Input: S3.TransactionType type Output: List<S3.Transaction> S3.Transaction DateTime Date decimal Amount string Description S3.TransactionType D,P client.ClientCredentials.UserName.UserName client.ClientCredentials.UserName.Password	• Missing method • Extra input field • Extra method • Missing output field
getPurchaseList Input: string accountCode Input: string password Output: List<S1.Transaction> S1.Transaction string AccCode DateTime Date decimal Amount string Description	getTransactions Input: DateTime fromDate Input: DateTime toDate Input: S3.TransactionType type Output: List<S3.Transaction> S3.Transaction DateTime Date decimal Amount string Description S3.TransactionType D,P client.ClientCredentials.UserName.UserName client.ClientCredentials.UserName.Password	• Missing method • Extra input field • Extra method • Missing output field
getAccountBalance Input: string accountCode Input: string password Output: decimal accountBalance	getAccountBalance Input: DateTime date Output: decimal accountBalance client.ClientCredentials.UserName.UserName client.ClientCredentials.UserName.Password	• Missing input field • Extra input field • Extra method

CONCLUSION AND FUTURE WORK

In this paper we propose the use of service adapters to tackle the problem of compatibility in different versions of a service. The main contributions of this work include (i) - proposing common mismatch types that can occur among different versions of a service (ii) - proposing a service adapter that can act as a service provider for the old service and a service consumer of the new service. We have shown that most of the incompatibilities that occur at the interface level are resolvable using our proposed method.

We have developed a proof-of-concept implementation of the proposed adapter. In particular, we have used the adapter to communicate two incompatible versions of a real world application.

Moreover, we did an interoperability test on WCF and Metro web services stacks. Our tests have shown that the Metro web services stack is not fully compatible with WCF. We have shown that some of the options in WCF have to be disabled to achieve interoperability.

Two main directions for future work are: developing a methodology that can generate the adapter automatically and enhancing the adapter to handle protocol mismatches.

References

[1]. *Service-oriented computing.* **Georgakopoulos, D., & Papazoglou, M. P.** 10, s.l. : Communications of the ACM - Service-oriented computing CACM , Oct 2003, Vol. 45, pp. 24-28.

[2]. **Judith Hurwitz, Robin Bloor,Carol Baroudi,Marcia Kaufman.** *Service Oriented Architecture For Dummies.* s.l. : Wiley Publishing, Inc., 2007.

[3]. **Microsoft.** Microsoft Application Architecture Guide 2nd Ed. s.l. : Microsoft, 2009.

[4]. **Berners-Lee, Tim.** Web Services. *The World Wide Web Consortium (W3C).* [Online] 2009. [Cited: Feb 4, 2011.] http://www.w3.org/DesignIssues/WebServices.html.

[5]. **Dirk Krafzig, Karl Banke, Dirk Slama.** *Enterprise SOA: Service-Oriented Architecture Best Practices.* s.l. : Prentice Hall PTR, 2004.

[6]. **Erl, Thomas.** *SOA Design Patterns.* s.l. : PRENTICE HALL, 2009.

[7]. **Mahmoud, Qusay H.** Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI). *ORACLE.* [Online] 2005. [Cited: Feb 8, 2011.] http://www.oracle.com/technetwork/articles/javase/soa-142870.html.

[8]. *Simple Object Access Protocol (SOAP) 1.1 : W3C Note 08 May 2000.* **Box, Don et al.** s.l. : DevelopMentor, International Business Machines Corporation, Lotus Development Corporation, Microsoft, UserLand Software, May 08, 2000.

[9]. **Hochgurtel, Brian.** *Cross-Platform Web Services Using C# and Java.* s.l. : Charles River Media, 2003.

[10]. **Graham, Steve et al.** *Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI.* s.l. : Sams Publishing, 2001.

[11]. **Wikipedea.** Web service. *Wikipedia.* [Online] [Cited: Feb 8, 2011.] http://en.wikipedia.org/wiki/Web_service.

[12]. **E. Gamma, R. Helm, R. Johnson, and J. Vlissides.** *Design patterns: elements of reusable object-oriented software.* s.l. : Addison-Wesley Longman Publishing, 1995.

[13]. **Wikipedia.** Adapter pattern. *Wikipedia.* [Online] Mar 16, 2011. http://en.wikipedia.org/wiki/Adapter_pattern.

[14]. *Discovery and Adaptation of Process Views.* **Nezhad, Hamid Reza Motahari.**

[15]. *Web Services Secure Conversation Language (WS-SecureConversation).* **Anderson, Steve et al.** s.l. : IBM, Microsoft and Actional, BEA, Computer Associates, Layer 7, Oblix, OpenNetwork, Ping Identity, Reactivity, and Verisign, Feb 2005, p. 2.

[16]. *Web Services Trust Language (WS-Trust).* **Anderson, Steve et al.** s.l. : Actional Corporation, BEA Systems, Inc., Computer Associates International, Inc., International Business Machines Corporation,and others, Feb 2005, p. 2.

[17]. **microsoft.** Messaging Specifications Index Page. *microsoft.com.* [Online] [Cited: Feb 11, 2011.] http://msdn.microsoft.com/en-us/library/ms951268.aspx.

[18]. *Web Services Reliable Messaging Protocol (WS-ReliableMessaging).* **Bilorusets, Ruslan et al.** s.l. : BEA Systems, IBM, Microsoft Corporation, Inc, and TIBCO Software Inc, Feb 2005, p. 2.

[19]. **Ferris, Chris.** First look at the WS-I Basic Profile 1.0. *ibm.com.* [Online] 2002. [Cited: Feb 11, 2011.] http://www.ibm.com/developerworks/webservices/library/ws-basicprof.html.

[20]. *Developing Adapters for Web Services Integration.* **B. Benatallah, F. Casati, D. Grigori, H. R.M. Nezhad, and F. Toumani.** In Proceedings of CAiSE'2005, pp. 415-429.

[21]. *Mismatch Patterns and Adaptation Aspects:A Foundation for Rapid Development of Web Service Adapters.* **Kongdenfha, Woralak, et al.** 2009, IEEE T. Services Computing (2009), pp. 94-107.

[22]. *Automated Generation of BPEL Adapters.* **Brogi, Antonio and Popescu, Razvan.** 2007, In Proceedings of CIbSE'2007, pp. 397-400.

[23]. *Interoperability among Independently Evolving Web Services.* **Ponnekanti, Shankar R. and Fox, Armando.** 2004, In Proceedings of Middleware'2004, pp. 331-351.

[24]. **Microsoft.** Metro to WCF Interoperability. *Microsoft.* [Online] [Cited: Mar 14, 2011.] http://msdn.microsoft.com/en-us/library/ff842400.aspx.

[25]. —. Windows Communication Foundation. *MSDN*. [Online] [Cited: Feb 19, 2011.] http://msdn.microsoft.com/en-us/library/ms735119%28v=vs.90%29.aspx.

[26]. **Wikipedia.** Windows Communication Foundation. *Wikipedia.* [Online] [Cited: Feb 21, 2011.] http://en.wikipedia.org/wiki/Windows_Communication_Foundation.

[27]. **Microsoft.** Fundamental Windows Communication Foundation Concepts. *Microsoft.* [Online] [Cited: Feb 19, 2011.] http://msdn.microsoft.com/en-us/library/ms731079.aspx.

[28]. **Cibraro, Pablo et al.** *Professional WCF 4.* s.l. : Wiley Publishing, 2010.

[29]. **Wikipedia.** Web Services Interoperability Technology. *Wikipedia.* [Online] [Cited: Feb 22, 2011.] http://en.wikipedia.org/wiki/Web_Services_Interoperability_Technology.

[30]. WSIT: Project Description. [Online] [Cited: Feb 22, 2011.] http://wsit.java.net/.

[31]. **Bustamante, Michèle Leroux.** Fundamentals of WCF Security. [Online] JANUARY 16, 2007. [Cited: Jan 22, 2011.] http://www.devx.com/codemag/Article/33342/1763/page/2.

[32]. **W3Schools.** Web Services Platform Elements. *W3Schools.* [Online] [Cited: Feb 8, 2011.] http://www.w3schools.com/webservices/ws_platform.asp.

