



Web Services Users Guide

Adaptive Server[®] Enterprise

15.7

DOCUMENT ID: DC10061-01-1570-02

LAST REVISED: March 2012

Copyright © 2012 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase trademarks can be viewed at the Sybase trademarks page at <http://www.sybase.com/detail?id=1011207>. Sybase and the marks listed are trademarks of Sybase, Inc. ® indicates registration in the United States of America.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

Java and all Java-based marks are trademarks or registered trademarks of Oracle and/or its affiliates in the U.S. and other countries.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

PART 1	INTRODUCTION	
CHAPTER 1	Understanding Adaptive Server Enterprise Web Services	3
	Overview	3
	Adaptive Server Enterprise Web Services	4
	Advantages of ASE Web Services	4
	Stored procedures and functions	4
	SQL	5
	Security	5
	LDAP	5
	User-defined Web services	5
	Web services standards	6
	XML	6
	WSDL	9
	SOAP	11
CHAPTER 2	Understanding the ASE Web Services Engine.....	13
	Producer of Web services	14
	Producer components	14
	Producer Web methods.....	15
	User-defined Web services	16
	Consumer of Web services	17
	Consumer components	17
	Proxy tables.....	19
	Web Services engine as a consumer versus producer	20
PART 2	CONFIGURATION	
CHAPTER 3	Configuring ASE Web Services	23
	Configuration	23
	Configuring during installation	24
	Configuring after installation	24

Licensing	25
Configuration files	25
ws.properties file	26
logging.properties file	26
wsmsg.properties file.....	26
Security	26
Configuring SSL	27
Installing a certificate for Microsoft .NET	28

PART 3 ADMINISTRATION

CHAPTER 4	Using ASE Web Services	33
	Using the ASE Web Services Engine	33
	Starting and stopping the ASE Web Services Engine.....	33
	ASE Web Services methods	36
	Using sp_webservices.....	39
	Invoking a Web service	43
	Using user-defined Web services	46
	Commands for user-defined Web services	46
	Using sp_webservices with user-defined Web services.....	51
	Security for user-defined Web services.....	54
	Auditing for user-defined Web services.....	54
	ASE Web Services logging	56
	ASE Web Services log files.....	56
	Rolling over log files	57
	Using Sybase Central	57

CHAPTER 5	Sample Applications	59
	Apache sample client	59
	Creating the sample client.....	59
	Using runexecute	60
	Microsoft .NET sample client	64
	Creating the sample client.....	64
	Using Execute.exe	64

PART 4 TROUBLESHOOTING

CHAPTER 6	Troubleshooting	69
	Troubleshooting issues	69
	Remote server class definition setting.....	69
	Unmapped RPC/encoded Web method.....	70

Truncated document/literal results	70
Starting ASE Web Services Engine	70
Locating WSDL	71
Specifying entries in ws.properties	71
Windows NT command-line arguments	72
Run or stop scripts fail	72
Null passwords	72
Specifying SOAP endpoints with SSL	73
Abnormal termination of sp_webservices 'add'	73
Web Services proxy table restrictions	73
sysattributes table entry	74
Diagnostic tools	74
Detailed logging	75
Enabling JDBC-level tracing	75
Messages	76

PART 5

APPENDIXES

APPENDIX A	Installation Contents	81
	ASE Web Services directory tree	81
	bin directory contents	82
	lib directory contents	82
	logs directory contents	83
	producer directory contents	83
	props directory contents	84
	samples directory contents	84
APPENDIX B	Configuration Properties	85
	ws.properties	85
	myres.properties	87
	Specifying properties file entries	89
APPENDIX C	SOAP and Adaptive Server Enterprise Datatype Mapping	91
	Datatype mapping	91
	SOAP-to-ASE datatype mappings	91
	ASE-to-SOAP datatype mappings for the create service command	94
		94
	Glossary	95
	Index	97

PART 1

Introduction

This part introduces Adaptive Server[®] Enterprise Web Services.

Understanding Adaptive Server Enterprise Web Services

This chapter discusses the following:

Topic	Page
Overview	3
Adaptive Server Enterprise Web Services	4
Advantages of ASE Web Services	4
Web services standards	6

Overview

A Web service is a self-contained, modular application that can be accessed through a network connection. Using a Web service, the end user trades performance for increased interoperability enforced by adherence to the Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Extensible Markup Language (XML) open standards.

Regardless of the programming language in which it has been implemented, a Web service can be accessed from many different platforms and operating systems, thus greatly enhancing the ability for diverse applications to share data. By using many discrete Web services, each handling a limited set of specific tasks, business enterprises can dynamically and incrementally integrate by exposing their existing software in a secure and controlled environment. By providing a standardized means to invoke remote applications, Web services reduce the amount of code required for infrastructure. By enabling users to extract implementation from exposed interfaces (WSDL), Web services provide the tools needed to build a service-oriented architecture (SOA).

Adaptive Server Enterprise Web Services

Adaptive Server Enterprise (ASE) Web Services consists of the ASE Web Services Engine that runs independently of Adaptive Server Enterprise.

The ASE Web Services Engine provides the following functionality:

- Enables client applications to access SQL and stored procedures in Adaptive Server Enterprise using SOAP.
- Enables Adaptive Server Enterprise to access the Web services of other applications. These external Web services are mapped to Adaptive Server Enterprise proxy tables at runtime.
- Provides user-defined Web services, which enable the execution of SQL commands in Adaptive Server Enterprise using a Web browser or SOAP client.

For more information on the ASE Web Services Engine, see Chapter 2, “Understanding the ASE Web Services Engine.”

Advantages of ASE Web Services

With the ASE Web Services Engine, the user can use stored procedures, user-defined functions, and SQL to query and manipulate data. A client application can send a SOAP request containing SQL commands and receive results through SOAP. Data is returned according to the SQLX standard, and the client application can receive XML data, schema, and DTDs.

Stored procedures and functions

Stored procedures separate the internal, logical view of the data from business-level logic and extend the influence and performance of SQL. Stored procedures can also be executed remotely. The user can use both stored procedures and user-defined functions to invoke Java methods, as specified in the ANSI SQLJ standard, and to retrieve data in standard XML format.

SQL

Because SQL can be used to manipulate XML data, SOAP-enabled client applications can use the ASE Web Services Engine to manage data in Adaptive Server Enterprise. SQL can also be used to invoke Web services through the ASE Web Services Engine.

Security

The ASE Web Services security features include Secure Sockets Layer (SSL) and provide important database security and authorization features, like access control through the Lightweight Directory Access Protocol (LDAP).

LDAP

LDAP is an Internet protocol for accessing directories in a distributed environment. An LDAP server stores the user information needed to establish connections between resources and grant access to directories, eliminating the need for client applications to know this information. ASE Web Services enables client applications to access Web methods using LDAP.

ASE Web Services supports LDAP version 3 servers. For more detailed information on using LDAP to enable user authentication and to locate Adaptive Server Enterprise data servers, see the *Security Administration Guide* for Adaptive Server Enterprise.

User-defined Web services

Using user-defined Web services, you can execute SQL commands in Adaptive Server Enterprise using a Web browser or SOAP client. This functionality allows you to define the name of a Web service, the SQL to execute, and the URL location.

User-defined Web services allow you to create an interface to Adaptive Server Enterprise that is compliant with SOA.

Web services standards

Web services are structured with XML, described with WSDL, and transferred with SOAP over HTTP. ASE Web Services enables client applications to access Web services and can consume remote Web services.

XML

XML is used to describe data. XML is derived from SGML and possesses some qualities of other markup languages, like HTML. However, XML is extensible because its tags are user-defined, making it ideal for exchanging data in a structure that is intelligible to two or more communicating applications.

Example

The following isql query to the pubs2 database finds information on discounts:

```
select * from discounts
```

This query produces the following result set:

discounttype	stor_id	lowqty	highqty	discount
Initial Customer	NULL	NULL	NULL	10.500000
Volume Discount	NULL	100	1000	6.700000
Huge Volume Discount	NULL	1001	NULL	10.000000
Customer Discount	8042	NULL	NULL	5.000000

This result set can be represented in XML in many ways. The following is an XML representation produced by ASE Web Services and formatted in SQLX, which is part of the ANSI standard for SQL:

```
<?xml version="1.0" encoding="UTF-8">
<ws xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <row>
    <discounttype>Initial Customer</discounttype>
    <discount>10.5</discount>
  </row>
  <row>
    <discounttype>Volume Discount</discounttype>
    <lowqty>100</lowqty>
    <highqty>1000</highqty>
```

```
    <discount>6.7</discount>
  </row>
  <row>
    <discounttype>Huge Volume Discount
  </discounttype>
    <lowqty>1001</lowqty>
    <discount>10.0</discount>
  </row>
  <row>
    <discounttype>Customer Discount</discounttype>
    <stor_id>8042</stor_id>
    <discount>5.0</discount>
  </row>
</ws>
```

The initial line describes the XML version and character encoding. The remaining tags are user-defined and describe both the structure and data of the document. These user-defined tags enable documents to be customized for a specific application, such as one that uses discount information to compute prices.

XML document structure

The user-defined elements and their arrangement in a well-formed XML document is defined either by a Document Type Definition (DTD) or an XML schema.

Following is a DTD for the previous example for discount information:

```
<!DOCTYPE ws [
  <!ELEMENT ws (row*)>
  <!ELEMENT row (discounttype, stor_id?, lowqty?,
  highqty?, discount)>
  <!ELEMENT discounttype (#PCDATA)>
  <!ELEMENT stor_id (#PCDATA)>
  <!ELEMENT lowqty (#PCDATA)>
  <!ELEMENT highqty (#PCDATA)>
  <!ELEMENT discount (#PCDATA)>
]>
```

The following is part of an XML schema for the previous example for discount information:

```
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:sqlxml="http://www.iso-
  standards.org/mra/9075/sqlx">
```

```
<xsd:import
  namespace="http://www.w3.org/2001/XMLSchema"
  schemaLocation="http://www.iso-
    standards.org/mra/9075/sqlx.xsd" />
<xsd:complexType name="RowType.ws">
  <xsd:sequence>
    <xsd:element name="discounttype"
      type="VARCHAR_40" />
    <xsd:element name="stor_id" type="CHAR_4"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="lowqty" type="SMALLINT"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="highqty" type="SMALLINT"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="discount" type="FLOAT" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TableType.ws">
  <xsd:sequence>
    <xsd:element name="row" type="RowType.ws"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="VARCHAR_40">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="40"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="VARCHAR_4">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="4"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="SMALLINT">
  <xsd:restriction base="xsd:integer">
    <xsd:maxInclusive value="32767"/>
    <xsd:minInclusive value="-32768"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="FLOAT">
  <xsd:restriction base="xsd:float"/>
</xsd:simpleType>
<xsd:element name="ws" type="TableType.ws"/>
</xsd:schema>
```

An XML schema or DTD can be included as part of the XML document they describe or be referenced as separate files. The respective file suffixes for an XML schema and a DTD are *.xsd* and *.dtd*.

For more detailed information on XML, refer to the following documents:

- World Wide Web Consortium (W3C), at <http://www.w3.org>
- W3C, Extensible Markup Language (XML), at <http://www.w3.org/XML/>

WSDL

A WSDL document is written in XML and describes a Web service. In addition to specifying the location of the Web service, a WSDL description also specifies the methods provided by the Web service, and the messages, datatypes, and communication protocols used by the Web service with the following tags:

- `<service>` – defines the name of the Web service. For example, a Web service called `ExecuteStoredProcService` could be named as follows:

```
<wsdl:service name="ExecuteStoredProcService">
  <wsdl:port binding="impl:aseSoapBinding" name="ase">
    <wsdlsoap:address location="http://myserver:8181/services/ase"/>
  </wsdl:port>
</wsdl:service>
```

A WSDL document may contain one or more `<service>` tags. In the case of the ASE Web Services Engine, there is only one service, which is named “ase.”

- `<binding>` – defines the communication protocols used. The following example uses the SOAP protocol:

```
<wsdl:binding name="aseSoapBinding" type="impl:ExecuteStoredProc">
...
</wsdl:binding>
```

WSDL also supports use of HTTP and MIME protocols.

- `<port>` – specifies the Web service address. For example:

```
<wsdl:port binding="impl:aseSoapBinding" name="ase">
  <wsdlsoap:address location="http://myserver:8181/services/ase"/>
</wsdl:port>
```

The `<port>` tag has attributes for name and binding.

- `<message>` – defines the messages used. For example:

```
<wsdl:message name="executeRequest">
  <wsdl:part name="service" type="xsd:string"/>
  <wsdl:part name="userName" type="xsd:string"/>
  <wsdl:part name="password" type="xsd:string"/>
  <wsdl:part name="sqlxOptions" type="xsd:string"/>
  <wsdl:part name="sql" type="xsd:string"/>
</wsdl:message>
```

This is a request message for a method called `executeRequest`. The `<part>` tags correspond to parameter values for the method call in a request message and to return values in a response.

- `<operation>` – associates a message with a Web method request or response. For example:

```
<wsdl:operation name="execute" parameterOrder="service userName
password sqlxOptions sql">
  <wsdl:input message="impl:executeRequest" name="executeRequest"/>
  <wsdl:output message="impl:executeResponse" name="executeResponse"/>
</wsdl:operation>
```

- `<portType>` – defines the methods provided. The `<operation>` tag is a child element of `<portType>`. For example:

```
<wsdl:portType name="ExecuteStoredProc">
  <wsdl:operation name="execute" parameterOrder="aseServerName
asePortNumber
....
  </wsdl:operation>
</wsdl:portType>
```

- `<types>` – defines the datatypes used. WSDL uses XML schema syntax to define datatypes.

WSDL is usually automatically generated by the ASE Web Services Engine and can be viewed in a Web browser at the following location:

`http://myserver:producer_port/services/ase?wsdl`

where *myserver* is the name of the host on which the ASE Web Services Engine is running, and *producer_port* is the port number.

SOAP

SOAP is a platform- and language-independent protocol based on XML that is used to send messages and data between applications. SOAP defines the structure of messages, describes how messages are to be processed, and provides rules for encoding application-defined datatypes. SOAP allows applications to send and receive remote procedure calls (RPCs) using any standard transport-layer protocol, usually HTTP.

SOAP message structure

A SOAP message consists of a header and a body, both of which are contained in a SOAP envelope. Generally, the SOAP request message contains no header information, but the response message corresponding to the previous request message contains a header and does not necessarily show the body of the message..

For more detailed information on SOAP, refer to the following documents:

- Simple Object Access Protocol (1.1):
<http://www.w3.org/TR/SOAP/>
- Simple Object Access Protocol (1.2) Part 1:
<http://www.w3.org/TR/soap12-part1/>
- Simple Object Access Protocol (1.2) Part 2:
<http://www.w3.org/TR/soap12-part2/>

Understanding the ASE Web Services Engine

This chapter discusses the following:

Topic	Page
Producer of Web services	14
Consumer of Web services	17

The ASE Web Services Engine provides the following functionality:

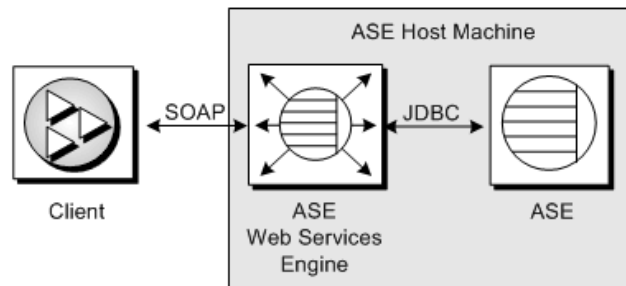
- Producer of Web services – enables a client application to access Adaptive Server Enterprise stored procedures and SQL using SOAP.
- Consumer of Web services – enables Adaptive Server Enterprise to access and execute Web methods.

Producer of Web services

The ASE Web Services Engine acts as a producer of Web services by enabling a client application to access Adaptive Server Enterprise stored procedures and SQL using SOAP. The output of the ASE Web Services Engine complies with SQLX, which is defined as part of the ANSI specification for SQL.

Note Sybase® recommends that you run the ASE Web Services Engine on the same machine as Adaptive Server Enterprise.

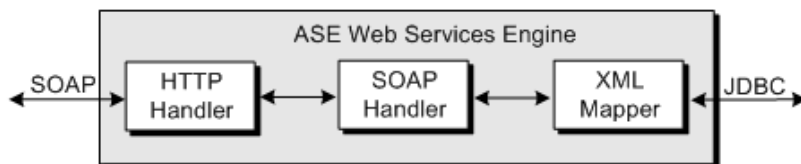
Figure 2-1: ASE Web Services Engine for client access to ASE



The client can send a SQL or stored procedure command as a SOAP request, and any result is returned as a SOAP response. The data in the SOAP response conforms to the SQLX standard.

Producer components

Acting as a producer of Web services, the ASE Web Services Engine uses three components: an HTTP handler, a SOAP handler, and an XML mapper.

Figure 2-2: ASE Web Services Engine as producer

HTTP handler

The HTTP handler supports HTTP 1.1 and listens for requests sent using the HTTP POST and GET methods. The HTTP handler also supports SSL connections.

Note Do not use GET HTTP requests. These commands embed all arguments within the URL, which cannot be encrypted. Use POST HTTP, which moves all arguments into the body of the HTTP request and allows the whole contents to be encrypted.

SOAP handler

The SOAP handler supports SOAP 1.2 and processes SOAP requests. The SOAP handler also generates WSDL files describing Web services.

XML mapper

The XML mapper encodes relational data, returned from Adaptive Server Enterprise through JDBC, into XML that complies with the SQLX standard. The XML mapper also generates a DTD and an XML schema to describe the data.

Producer Web methods

The ASE Web Services Engine provides the following methods:

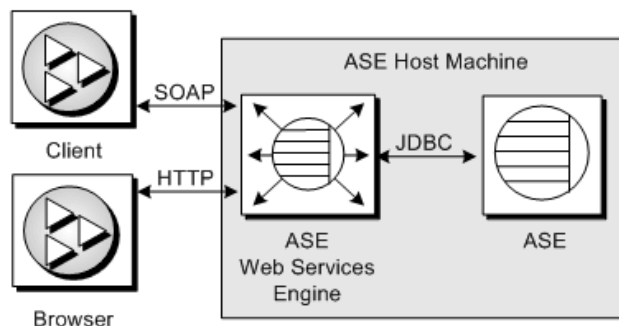
- execute – executes a SQL statement or stored procedure.
- login – establishes a persistent connection to Adaptive Server Enterprise.
- logout – explicitly terminates an Adaptive Server Enterprise connection.

For information on using these Web methods, see Chapter 4, “Using ASE Web Services.”

User-defined Web services

In addition to the Web methods provided by the ASE Web Services Engine, ASE Web Services enables you to create Web services and execute SQL commands in Adaptive Server Enterprise using either a Web browser or a SOAP client.

Figure 2-3: User-defined Web services



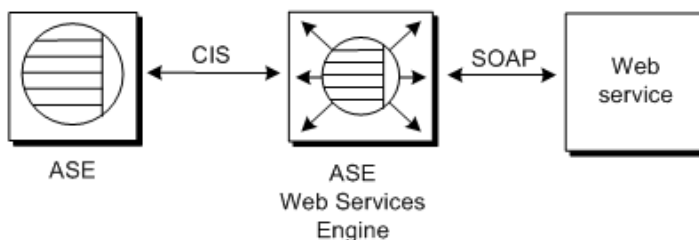
You can create a user-defined Web service with the create service command, which enables you to specify the SQL to be executed, create a first-class object for which permissions can be controlled with the grant command, and control whether the service can be invoked with a Web browser or a SOAP client. The ASE Web Services Engine automatically generates WSDL for user-defined Web services. For details on creating and using user-defined Web services, see Chapter 4, “Using ASE Web Services.”

Note Do not use GET HTTP requests. These commands embed all arguments within the URL, which cannot be encrypted. Use POST HTTP, which moves all arguments into the body of the HTTP request and allows the whole contents to be encrypted.

Consumer of Web services

The ASE Web Services Engine acts as a consumer of Web services by enabling Adaptive Server Enterprise to access and execute Web methods. A Web method is made accessible by mapping it to an Adaptive Server Enterprise proxy table using information provided in the WSDL file for the Web method. A Web method can then be invoked with a select on the proxy table.

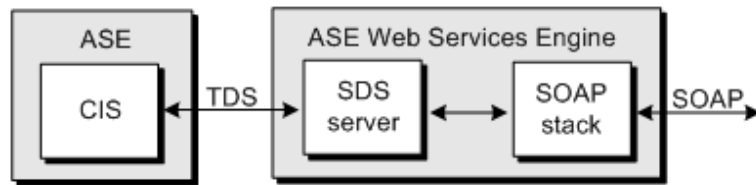
Figure 2-4: Accessing remote Web services



Note The Web service accessed may reside within or beyond a firewall.

Consumer components

Acting as a consumer of Web services, the ASE Web Services Engine uses a Specialty Data Store (SDS) server and a WSDL parser. The SDS is used as a Component Integration Service (CIS) to map the Web method to a proxy table. The proxy table is constructed using a WSDL file parsed with Apache Axis.

Figure 2-5: ASE Web Services Engine as consumer

SDS server

When Adaptive Server Enterprise receives a select statement for a Web method proxy table, Adaptive Server Enterprise forwards the request to the ASE Web Services Engine in a Tabular Data Stream™ (TDS). The SDS server, which acts as a server for CIS, allows the ASE Web Services Engine to intercept and handle the TDS from Adaptive Server Enterprise.

SOAP stack

The SOAP stack is a layered set of functionalities that collectively handle the serialization and transport of XML-encoded data. The SOAP stack uses the WSDL file for a Web method to determine the structure of a corresponding proxy table. The SOAP stack also generates SOAP requests corresponding to the select statement submitted to Adaptive Server Enterprise and sends these SOAP requests to a SOAP server. The SOAP stack supports both RPC/encoded and document/literal Web methods.

RPC/encoded methods

The SOAP messages for an RPC/encoded Web method contain an XML element for each method parameter. Messages for an RPC/encoded Web method are encoded according to the SOAP specification. The proxy table representing an RPC/encoded Web method contains a column for each input and output parameter.

Note If an RPC/encoded Web method has no input or output parameters, it cannot be mapped to a proxy table. A proxy table for a Web method without parameters would have no columns. A table with no columns cannot be created in Adaptive Server Enterprise.

Note Currently, only simple types are mapped to columns. Complex types or arrays used in RPC/encoded Web methods result in the Web method not being mapped to a proxy table.

Document/literal methods

In the SOAP messages for a document/literal Web method, the communicating parties specify the data being transmitted and formatted according to XML schemas incorporated into the WSDL file. Messages for a document/literal Web method are serialized and deserialized according to the WSDL file for the Web method. The proxy table representing a document/literal Web method contains two columns: `_inxml` and `outxml`.

The Web Services Interoperability (WSI) Organization, which defines practical interoperability guidelines, recommends using document/literal Web methods to enhance portability. Sybase supports this recommendation.

Proxy tables

Because they point to Web methods, Web Services proxy tables are of type procedure and are subject to all restrictions for this type:

- *Commands* – You cannot issue a delete, update, or insert command against a table of type procedure, nor can you issue a create index, truncate table, or alter table command.
- *Joins* – A Web Services proxy table can only be joined with one other table, and that table cannot be another Web Services proxy table.
- *Queries* – Column names beginning with underscore ('_') are used to specify input parameters. These columns are referred to as parameter columns and must be in the where clause of a select statement.

For a complete description of the restrictions for procedure tables, see the *Component Integration Services Users Guide*.

Web Services engine as a consumer versus producer

When the Web Services engine acts as a consumer of Web Services (that is, when Adaptive Server queries Web Services for the results of web service calls from any external services available on the Internet), Adaptive Server can make requests from any Web Services engine that is defined in its `syssservers` table.

When the Web Services engine acts as a producer of Web Services defined on Adaptive Server, the Web Services engine is aware only of the Adaptive Server defined in its properties file, and can request results from only this Adaptive Server.

PART 2

Configuration

This part provides instructions for configuring Web Services in Adaptive Server.



Configuring ASE Web Services

This chapter discusses the following:

Topic	Page
Configuration	23
Licensing	25
Configuration files	25
Security	26

ASE Web Services is installed as part of the installation for Adaptive Server Enterprise. ASE Web Services may be configured during the installation of Adaptive Server Enterprise using a configuration wizard or after the installation from Sybase Central. For instructions on installing Adaptive Server Enterprise, see the *Installation Guide* for Adaptive Server Enterprise.

Note Unless otherwise specified, directories listed in this and subsequent chapters are assumed to reside under the `$SYBASE/WS-15_0` directory on UNIX and the `%SYBASE%\WS-15_0` directory on Windows.

Configuration

You can use the Configuration Utility to configure ASE Web Services either during or after the installation of ASE Web Services:

- Configuring during installation – You have the option of configuring ASE Web Services during a custom installation of the ASE Web Services feature.

- Configuring after installation – If you do not configure the Web Services feature during installation, you must configure the feature from Sybase Central after the installation is complete. Sybase Central provides a GUI Configuration Utility similar to that used in the installation procedure. You can also configure the Web Services feature in silent mode or in command/console mode.

Configuring during installation

To configure Web Services, activate the Configuration Utility during the installation of the Web Services feature, and follow the steps in the wizard.

Configuring after installation

If you have already installed ASE Web Services, you can start the Configuration Utility from the Adaptive Server Enterprise plug-in to Sybase Central. You can configure Web Services using the installation GUI, or you can use the command/console or silent modes.

❖ Starting the Configuration Utility GUI from Sybase Central

- 1 Click on the Utilities folder in the Folders view for the Adaptive Server Enterprise plug-in of Sybase Central.
- 2 Find the Configure Web Service icon in the details view to the right of the Folders view. Select the icon for Configure Web Service to start the Configuration Utility.
- 3 Follow the steps in the wizard.

❖ Starting the Configuration Utility in command/console mode

- 1 Open a console window.
- 2 Enter the following command:

```
aseplugin -I
```
- 3 Follow the steps indicated on the console.

❖ Starting the Configuration Utility in silent mode

- 1 Edit the entries in your *myres.properties* file to indicate the desired values. To set a property, add “=” and the property value to the *myres.properties* entry. For details on the contents of the *myres.properties* file, see “myres.properties” in the Appendix, “Configuration Properties.”
- 2 Open a console window.
- 3 Enter the following command:

```
aseplugin -s path
```

where *path* is the path to your *myres.properties* file.

In silent mode, no further user action is necessary to configure Web Services.

Licensing

License entry for ASE Web Services is handled by the InstallShield installation for Adaptive Server Enterprise.

To check out your ASE Web Services license from SySAM, do the following before you run the ASE Web Services Engine:

- 1 Establish an isql session with your Adaptive Server Enterprise.
- 2 Activate ASE Web Services by entering the following command in isql:

```
sp_configure 'enable webservices', 1
```

Configuration files

The *props* directory contains the following configuration files:

- ws.properties file
- logging.properties file
- wsmg.properties file

ws.properties file

The *ws.properties* file contains configuration settings for ASE Web Services. For details on these configuration properties, see Appendix B, “Configuration Properties.”

logging.properties file

The *logging.properties* file defines where logging output is sent. You can direct output to a log file or to the console. If output is directed to a log file, you can specify when to roll over to a new log file. You can also control the format of logging messages. For information on log rollover policies, see “Rolling over log files” on page 57.

The *logging.properties* file entries and logging behavior follow that for the Apache log4j package. For detailed information, refer to the documentation for log4j at <http://jakarta.apache.org/log4j/docs/documentation.html>.

wsmmsg.properties file

The *wsmmsg.properties* file is for internal use by ASE Web Services.

Security

To ensure secure operation of ASE Web Services, Sybase recommends that you do the following:

- Install ASE Web Services on the same machine as Adaptive Server Enterprise.
- Use SSL to connect to the ASE Web Services Engine. For instructions on configuring SSL, see “Configuring SSL” on page 27.

ASE Web Services supports all authorization measures supported by Adaptive Server Enterprise.

Note For the most current information on security in ASE Web Services, see the Adaptive Server Enterprise *Release Bulletin*.

Configuring SSL

Note Two certificate passwords are created by default as “sybase” from InstallShield during installation. You can change these from Sybase Central later.

SSL is configured automatically using the Configuration Utility from InstallShield or from Sybase Central. However, you can also configure SSL manually. To manually configure SSL for ASE Web Services, run the *configssl* script, which can be found in the *bin* directory:

```
configssl -d <domain_hostName> -k <keystore>
-h <httpsPort> -f <property_file>
-c <certificate_password> -s <keystore_password>
```

where:

- *domain_hostName* is the host name of the URL to connect to using SSL. For example, the *domain_hostName* for the following URL would be *mydomainhostname*:

```
http://mydomainhostname:8183/services/ase
```

There is no default for this parameter value.

- *keystore* is the location and file at which to store certificates. The default location for UNIX is *\$SYBASE/WS-15_0/props/keystore*, or *%SYBASE%\WS-15_0\props\keystore* for Windows.
- *httpsPort* is the port on which to listen for an SSL connection. The default is 8182.
- *property_file* is the location and name of the properties file to update. The default location for UNIX is *\$SYBASE/WS-15_0/props/ws.properties*, or *%SYBASE%\WS-15_0\props\ws.properties* for Windows.
- *certificate_password* is the password for the certificate. There is no default for this parameter value. If no password is supplied when the script is invoked, the script will prompt for a value.

- *keystore_password* is the password for the keystore. There is no default for this parameter value. If no password is supplied when the script is invoked, the script will prompt for a value.

Note You can also add your own certificate for SSL. For instructions on how to add your own certificate, see the documentation for the *keytool* utility in the JRE that manipulates the *keystore* file. The JRE delivered with ASE Web Services is version 1.4.

The *WS.properties* file sets the *keystore* location with *com.sybase.ase.ws.producer.ssl.keystore*. The WS Producer is the only user of the *keystore* location. The WS Consumer uses a separate certificate, which is located in the JRE by default. To consume Webservices over HTTPS (SSL) as a WS consumer, import the Producer's SSL certificate into the keystore for the JRE (for example, with *\$SYBASE/\$SYBASE_JRE6_64/lib/security/cacerts*

Installing a certificate for Microsoft .NET

A Microsoft .NET client requires a certificate to access the ASE Web Services Engine using SSL. Use the following procedure to install a certificate for Microsoft .NET.

❖ Installing a certificate for Microsoft .NET

- 1 Start the ASE Web Services Engine with SSL. For instructions on starting the ASE Web Services Engine, see “Starting and stopping the ASE Web Services Engine” on page 33.

- 2 Enter the following in the Address bar of Microsoft Internet Explorer:

`https://<producer_host>:<SSL_port>`

where:

- *producer_host* is the host on which the ASE Web Services Engine runs.
- *SSL_port* is the port for the ASE Web Services Engine.

The Security Alert dialog box appears.

- 3 Click View Certificate. The Certificate dialog box appears.
- 4 Click Install Certificate. The Certificate Manager Import wizard opens.

- 5 Click Next until the Certificate Manager Import wizard indicates that the certificate was successfully installed and returns to the Certificate dialog box.
- 6 Click OK. The browser returns you to the Security Alert dialog box.
- 7 Click Yes. The browser window should display a page titled “Welcome to the ASE Web Services.”

❖ **Verifying the certificate installation**

- 1 Close all browser windows.
- 2 Restart Microsoft Internet Explorer.
- 3 Enter the following in the Address bar of Microsoft Internet Explorer:

`https://<producer_host>:<SSL_port>`

where:

- *producer_host* is the host on which the ASE Web Services Engine runs.
- *SSL_port* is the port for the ASE Web Services Engine.

No Security Alert dialog box should appear.

PART 3

Administration

This part describes product functionality and administrative tasks.



Using ASE Web Services

This chapter discusses the following:

Topic	Page
Using the ASE Web Services Engine	33
Using user-defined Web services	46
ASE Web Services logging	56
Using Sybase Central	57

Before using ASE Web Services, make sure you have completed the configuration tasks in Chapter 3, “Configuring ASE Web Services.”

Using the ASE Web Services Engine

This section documents the following:

- Starting and stopping the ASE Web Services Engine
- ASE Web Services methods
- Using `sp_webservices`
- Invoking a Web service

Starting and stopping the ASE Web Services Engine

To start the ASE Web Services Engine for ASE Web Services, execute the `runws` script, which is located in the `bin` directory:

```
runws -U <ase_username> -P <ase_password>
-S <ase_server_name> -f <property_file> -v
```

To stop the ASE Web Services Engine for ASE Web Services, execute the `stopws` script, also located in the `bin` directory:

```
stopws -U <ase_username> -P <ase_password>
```

```
-S <ase_server_name> -f <property_file> -v
```

The parameters are the same for both the *runws* and *stopws* scripts:

- *ase_username* is the user name for the Adaptive Server Enterprise. There is no default for this parameter value. If you do not supply a value for this parameter, you will be prompted for one.
- *ase_password* is the password for the Adaptive Server Enterprise. There is no default for this parameter value. If you do not supply a value for this parameter, you will be prompted for one.
- *ase_server_name* is the name of the Web service. There is no default for this parameter value. If you do not supply a value for this parameter, you will be prompted for one.
- *property_file* is the location and name of the properties file to update. The default location for UNIX is `SYBASE/WS-15_0/props/ws.properties`, or `%SYBASE%\WS-15_0\props\ws.properties` for Windows.
- `-v` specifies that the ASE Web Services Engine should display version information at startup or shutdown.

Conditions

The ASE Web Services Engine will start or stop if the following conditions are met:

- The *ase_server_name* provided is located on an LDAP server pointed to by the *libtcl.cfg* file or in the *interfaces* file for Adaptive Server Enterprise. ASE Web Services first searches for an entry containing the value of *ase_server_name* on an LDAP server pointed to by the *libtcl.cfg* file. ASE Web Services locates the *libtcl.cfg* file using the `com.sybase.ase.ws.libtcl` entry in the *ws.properties* file. If no entry is found on an LDAP server, ASE Web Services looks for an entry in the *interfaces* file for Adaptive Server Enterprise.

Note On Windows systems, the *interfaces* file is named *sql.ini*.

ASE Web Services locates the *interfaces* file using the `com.sybase.ase.ws.interfaces` entry in the *ws.properties* file.

- A successful login can be made using the *ase_username* and *ase_password* provided.

Note The password for an Adaptive Server Enterprise user cannot be set to a null string. The sa login allows null-string passwords by default. Sybase does not recommend using null passwords.

- The login to Adaptive Server Enterprise has sa role privileges.
- The following stored procedure command has been executed in isql for your Adaptive Server Enterprise:

```
sp_configure 'enable webservices', 1
```

Verification

After successfully executing the *runws* script, verify that ASE Web Services is enabled and that the ASE Web Services Engine is running.

❖ Verifying that ASE Web Services is enabled

To verify that ASE Web Services is enabled:

- Execute the following command on Adaptive Server Enterprise:

```
sp_configure 'enable webservices'
```

If *sp_configure* returns a value of 1, the Web Services feature has been enabled. A return value of 0 indicates the feature is not enabled.

❖ Verifying that the ASE Web Services Engine is running

- Check the *producer.log* or *consumer.log* file in the *logs* directory for messages indicating that the ASE Web Services Engine is running. For example:

```
2004-03-29 16:29:29.522 INFO [main] - Starting HTTP Server on Port: 8181
```

For SSL, the log indicates an HTTPS port and related SSL information. For example:

```
2004-03-29 16:29:29.532 INFO [main] - Https Port [8182], KeyPassword: ...
```

Note The *runproducer*, *stopproducer*, *runconsumer*, and *stopconsumer* scripts have been kept in 15.0 release for compatibility with previous releases of ASE Web Services. However, in the 15.0 release, these scripts call the *runws* and *stopws* scripts.

ASE Web Services methods

To access ASE Web Services, your client must use the methods exposed by the ASE Web Services Engine. These methods are mapped in SOAP as rpc:

```
<soap:binding style="rpc" ...>
```

Message data is encoded:

```
<soap:body use="encoded" ...>
```

The ASE Web Services Engine provides the following methods:

- `execute` – executes a SQL statement or stored procedure.
- `login` – establishes a persistent connection to Adaptive Server Enterprise.
- `logout` – explicitly terminates an Adaptive Server Enterprise connection.

These methods are supported by Adaptive Server Enterprise by default and are provided as one Web service (with one WSDL file). The syntax for these methods is the same whether they are invoked using HTTP or SSL.

execute

The `execute` method executes a Transact-SQL statement or stored procedure in Adaptive Server Enterprise.

Syntax

```
execute aseServerName userName password sqlxOptions sql
```

Parameters

- *aseServerName*
SOAP string indicating the name of the Adaptive Server Enterprise server in the *interfaces* file or LDAP server.

At each invocation of the `execute` method, ASE Web Services uses the value of *aseServerName* in the same way that *ase_server_name* is used to start or stop the ASE Web Services Engine. See “Starting and stopping the ASE Web Services Engine” on page 33 for details.
- *userName*
SOAP string indicating the user ID needed to log in to the Adaptive Server Enterprise.
- *password*
SOAP string indicating the password needed to log in to the Adaptive Server Enterprise.
- *sqlxOptions*

SOAP string indicating one or more option parameters. These parameters specify characteristics of the SQLX result set. The following are valid option parameters:

- general
- binary={hex | base64}
- columnstyle={element | attribute}
- entitize={yes | no | cond}
- format={yes | no}
- header={yes | no}
- multipleentitize={yes | no}
- multipleresults={all | data}
- ncr={non_ascii | no}
- nullstyle={attribute | omit}
- prefix="value"
- root={yes | no}
- rowname="value"
- schemaloc="value"
- statement={yes | no}
- tablename="value"
- targetns="value"
- xsidecl={yes | no}

You must provide a value for *value*. For more information on SQLX functions and options, see *XML Services in Adaptive Server Enterprise*.

- *sql*

SOAP string indicating the SQL statement or stored procedure to be executed on Adaptive Server Enterprise. The size of the SOAP string specified in the *sql* parameter is limited by the value of the `com.sybase.ase.ws.maxpostsize` property setting in the `ws.properties` file. For a description of this and other properties, see Appendix B, "Configuration Properties."

Example 1

This example checks the version number for Adaptive Server Enterprise.

```
execute johndoe-sun sa password "tablename=ws" "select
@@version"
```

This example invokes the Web method directly. ASE Web Services returns an XML schema, a DTD, and a result set containing the result of the executed statement.

Note You must provide the *userName* and password parameters to invoke the execute method. Adaptive Server verifies it can execute the specified SQL statement.

Example 2

This example computes a left join on tables in the pubs2 database.

```
execute johndoe-sun sa password "tablename=ws"
"select title, price, au_fname, au_lname from (titles
left join titleauthor on titles.title_id =
titleauthor.title_id ) left join authors on
titleauthor.au_id = authors.au_id and titles.price >
$15.00"
```

login

The login method establishes a persistent connection to Adaptive Server Enterprise.

Syntax

```
login aseServerName userName password
```

Parameters

- *aseServerName*
SOAP string indicating the name of the Adaptive Server Enterprise on which to execute the SQL statement or stored procedure.
For the login method, ASE Web Services uses the value of *aseServerName* in the same manner as for the execute method.
- *username*
SOAP string indicating the user ID needed to log in to the Adaptive Server Enterprise.
- *password*
SOAP string indicating the password needed to log in to the Adaptive Server Enterprise.

Usage Before a SQL statement or stored procedure can be executed on Adaptive Server Enterprise, a connection must first be established. However, the login method is optional. If you invoke an execute method without first invoking the login method, ASE Web Services automatically establishes a non-persistent connection to Adaptive Server Enterprise. The login method initiates a persistent connection to Adaptive Server Enterprise. The connection is terminated with the logout method. Persistent connections that are inactive for 60 seconds are terminated automatically.

logout

The logout method terminates a persistent connection to Adaptive Server Enterprise.

Syntax `logout`

Usage The logout method terminates a persistent connection to Adaptive Server Enterprise established by the login method.

Using sp_webservices

The `sp_webservices` stored procedure creates and manages the proxy tables used in the ASE Web Services Engine. This section documents the options and parameters for `sp_webservices`.

The `sp_webservices` stored procedure has the following options:

- `add` – creates a proxy table.
- `help` – displays usage information for `sp_webservices`.
- `list` – lists the proxy tables mapped to a WSDL file.
- `modify` – modifies timeout setting.
- `remove` – removes proxy tables mapped to a WSDL file.

There are additional `sp_webservices` options for use with user-defined Web services. For information on these options, see “Using `sp_webservices` with user-defined Web services” on page 51.

Note For information on restrictions for Web Services proxy tables, see “Web Services proxy table restrictions” in Chapter 6, “Troubleshooting.”

add

The add option is used to create a proxy table for a Web method specified by a WSDL file. When the add option is used successfully, the list option is invoked automatically to describe the schema of the new proxy table.

Syntax

```
sp_webservices 'add', 'wsdl_uri' [, sds_name]
    [, 'method_name=proxy_table
    [,method_name=proxy_table]* ' ]
```

Parameters

- *wsdl_uri*
The location for the WSDL file to be mapped to the new proxy table. If this parameter is specified, Web Services ensures that the URI exists in the syswsdl table.
- *sds_name*
The name specified for the ASE Web Services Engine in the *interfaces* or *sql.ini* file. The default value is ws. If no entry exists in the sysattributes table, an error results.
- *method_name*
The name of the Web method to be mapped to a proxy table. The *method_name* specified must be the name of a Web method specified in the associated WSDL file.
- *proxy_table*
The name of proxy table to which the Web method specified in *method_name* is mapped.

Usage

If you not specify *method_name* and *proxy_table* values for a Web method, the proxy table generated for the Web method is, by default, the name of the Web method specified in the WSDL file. If there is already a proxy table with the name of this Web method, a new proxy table is generated with a name like the following:

```
method_nameN
```

where *method_name* is the default proxy table name, and *N* is a digit from 1 to 9 denoting each successive mapping of the Web method. There can be as many as 99 duplicate proxy tables.

If you do specify *method_name* and *proxy_table* values for a Web method, the name of the proxy table must be new. If there is already a proxy table with the name specified in *proxy_table*, an error results, and none of the Web methods specified in the add option are mapped to proxy tables.

The output from the add option lists the methods that have been successfully mapped to proxy tables as well as those that have not been mapped. The name of a proxy table for an unmapped Web method is indicated as NULL in the output from the add option.

Note The columns used for input and output vary for proxy tables generated for RPC/encoded Web methods and document/literal Web methods. A proxy table representing an RPC/encoded Web method contains a column for each input and output parameter. A proxy table representing a document/literal Web method contains two columns, `_inxml` and `outxml`.

Note For information on datatype mappings, see Appendix C, “SOAP and Adaptive Server Enterprise Datatype Mapping.”

help

The help option provides instructions and examples illustrating how to use the `sp_webservices` stored procedure.

Syntax

```
sp_webservices help [, 'option']
```

Parameters

- *option*

The option for which to provide detailed instructions. Valid values are `add`, `list`, `remove`, and `modify`.

Usage

If no value is specified for *option*, the help option prints a brief syntax description for the `add`, `addalias`, `deploy`, `dropalias`, `list`, `listalias`, `listudws`, `modify`, `remove`, and `undeploy` options.

list

The list option is used to list Web methods described in a WSDL file.

Syntax

```
sp_webservices 'list' [, 'wsdl_uri'] [, sds_name]
```

Parameters

- *wsdl_uri*

The URI for the mapped WSDL file. If no value is specified for *wsdl_uri*, the list option displays information about all Web methods that have been mapped to proxy tables.

- *sds_name*

The name of the SDS server specified for the ASE Web Services Engine in the *interfaces* or *sql.ini* file. The default value is *ws*. If no entry exists in the *sysattributes* table, an error results.

If neither the *wSDL_uri* nor the *sds_name* parameter is specified, all entries in the *sysattributes* table are listed, ordered by *wSDLid*.

Usage

If the Web methods described in the WSDL file have already been mapped to proxy tables, the *list* option prints information about each proxy table. If the Web methods described in the WSDL file have not already been mapped to proxy tables, the *list* option prints SQL that can be used to create proxy tables.

modify

The *modify* option is used to modify the attribute information for a WSDL file.

Syntax

```
sp_webservices 'modify', 'wSDL_uri', 'timeout=time'
```

Parameters

- *wSDL_uri*

The URI of the WSDL file for which attribute information is to be changed.

- *time*

The interval in seconds during which a Web method must respond before the operation is aborted.

remove

The *remove* option is used to remove a proxy table mapping for a Web method.

Syntax

```
sp_webservices 'remove', 'wSDL_uri' [, sds_name]
```

Parameters

- *wSDL_uri*

The URI of the WSDL file for which the proxy table is to be removed.

- *sds_name*

The name of the SDS server specified for the ASE Web Services Engine in the *interfaces* or *sql.ini* file. The default value is *ws*. If no entry exists in the *sysattributes* table, an error results.

Invoking a Web service

To invoke a Web service using the ASE Web Services Engine, use the following procedure:

❖ Invoking a Web service

- 1 Start the ASE Web Services Engine.
- 2 Use the add option of `sp_webservices` to map the Web service to a proxy table in Adaptive Server Enterprise.
- 3 Use `sp_help` to determine the input and output parameters needed to invoke the Web method.
- 4 Invoke the Web method with a select statement on the proxy table.

Example 1

This example invokes an RPC/encoded Web method to display the exchange rate between two currencies.

Use the add option of `sp_webservices` to map Web methods to proxy tables:

```
1> sp_webservices 'add',
'http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl'
2> go
```

The Web method `getRate` is mapped to a proxy table of the same name.

Invoke the Web method by selecting from the proxy table:

```
1> select * from getRate where _country1 = 'usa' and _country2 = 'india'
2> go
```

The results returned for the previous select show the exchange rate for the specified parameters:

```
Result          _country1      _country2
43.000000      usa            india
(1 row affected)
```

Example 2

This example invokes a Web method to display stock information within an XML document.

Use the add option of sp_webservices to map Web methods to proxy tables:

```
1> sp_webservices "add" , "http://www.webserviceX.net/stockquote.asmx?WSDL"
2> go
```

The Web method GetQuote1 is mapped to a proxy table of the same name.

Invoke the Web method by selecting the outxml column of the GetQuote1 proxy table:

```
1> select outxml from GetQuote1 where _inxml = '<?xml version="1.0"
encoding="utf-8"?>
2>     <GetQuote1 xmlns="http://www.webserviceX.NET/">
3>         <symbol>SY</symbol>
4>     </GetQuote1>'
5> go
```

The results for the previous select display quote information within an XML document:

outxml

```
<?xml version="1.0" encoding="UTF-8" ?><GetQuote1Response
xmlns="http://www.webserviceX.NET/"><GetQuoteResult><StockQuotes><Stock><Symbol>SY</Symbol><Last>21.48</Last><Date>7/21/2005</Date><Time>4:01pm</Time><Change>+1.72</Change><Open>20.00</Open><High>21.60</High><Low>19.91</Low><Volume>2420100</Volume><MktCap>1.927B</MktCap><PreviousClose>19.76</PreviousClose><PercentageChange>+8.70%</PercentageChange><AnnRange>12.75 - 20.44</AnnRange><Earnings>0.706</Earnings><P-E>27.99</P-E><Name>SYBASE INC</Name></Stock></StockQuotes></GetQuoteResult></GetQuote1Response>
```

(1 row affected)

Example 3

This example invokes the GetQuote1 Web method, mapped to a proxy table in the previous example, through a view to display stock information.

To use this Web service, you must create a table to hold symbols representing stocks:

```
1> create table stocksymbol(symbol varchar(100))
2> go
```

Insert data into the stocksymbol table:

```

1> insert stocksymbol values("SY")
2> insert stocksymbol values("ORCL")
3> go

```

Now create a view that invokes the GetQuote1 Web method:

```

1> CREATE VIEW getstockvw as
2> select Symbol = xmlextract('//Stock/Symbol/text()',outxml returns
varchar(5)),
3>   Name = xmlextract('//Stock/Name/text()',outxml returns varchar(20)),
4>   Time = xmlextract('//Stock/Time/text()',outxml returns varchar(10)),
5>   Date = xmlextract('//Stock/Date/text()',outxml returns date),
6>   High = xmlextract('//Stock/High/text()',outxml returns decimal(15,2)),
7>   Low = xmlextract('//Stock/Low/text()',outxml returns decimal(15,2))
8> FROM GetQuote1 ,stocksymboll
9> WHERE _inxml = '<GetQuote1
xmlns="http://www.webserviceX.NET/"><symbol>'+symbol+'</symbol></GetQuote1>'
10> go

```

Select from the getstockvw view to view output from the GetQuote1 method:

```

1> select * from getstockvw
2> go

```

The results for the previous select display quote information for the parameters specified by the view definition:

Symbol	Name	Time	Date	High	Low
SY	SYBASE INC	4:01pm	Jul 21 2005	21.60	19.91
ORCL	ORACLE CORP	4:00pm	Jul 21 2005	14.05	13.54
MSFT	MICROSOFT CP	4:00pm	Jul 21 2005	26.48	26.19

(3 rows affected)

Using user-defined Web services

This section describes functionality specific to user-defined Web services and covers the following topics:

- Commands for user-defined Web services
- Using `sp_webservices` with user-defined Web services
- Security for user-defined Web services
- Auditing for user-defined Web services

Commands for user-defined Web services

You can create, drop, and alter user-defined Web services using the following commands:

- `create service`
- `drop service`

create service

The `create service` command wraps the supplied SQL statement in a stored procedure with the specified name and parameters. Except for the following differences, the resulting stored procedure behaves the same as a stored procedure created with the `create procedure` command, follows existing stored procedure rules for execution, replication, `sp_helptext`, and recompilation, and is executable from `isql`:

- The resulting stored procedure can be dropped only with the `drop service` command, not the `drop procedure` command.
- The `syscomments` table is populated with DDL necessary to recreate the `create service` command.
- The specified service name may not create a stored procedure group.

Note To make a user-defined Web service available through the ASE Web Services Engine, you must use the `deploy` option of `sp_webservices`. However, the stored procedure for a user-defined Web service is accessible from `isql`, even if it has not been deployed. For information on the `deploy` option of `sp_webservices`, see “`deploy`” on page 52 in “Using `sp_webservices`.”

Syntax	<pre> create service <i>service-name</i> [secure <i>security_options</i>] [, userpath <i>path</i>] [, alias <i>alias-name</i>] type { xml raw soap } [[(@<i>parameter_name</i> datatype [(length) (precision [, scale])] [= default][output] [, @<i>parameter_name</i> datatype [(length) (precision [, scale])] [= default][output]...)]]] as <i>SQL_statements</i> <i>security_options</i> ::= (<i>security_option_item</i> [<i>security_option_item</i>]) </pre>
Parameters	<ul style="list-style-type: none"> • <i>service-name</i> – the name for the user-defined Web service. This name can be any name that is valid for a stored procedure. When the drop service command is invoked with this service name, the corresponding stored procedure is dropped. If you specify the name of an existing service, an exception results. • <i>security_option_item</i> – either clear or ssl: <ul style="list-style-type: none"> • clear indicates that HTTP is used to access this Web service. • ssl indicates HTTPS is used to access this Web service • <i>path</i> – a character-string literal specifying the user-defined path to be appended to the URL accessing the Web service. This path is null by default. • <i>alias-name</i> – a character-string-literal specifying the user-defined Web service alias. • <i>parameter_name</i> – the name of an argument to the user-defined Web service. The value of this parameter is supplied when the Web service is executed. Parameter names must be preceded by the @ sign and conform to the rules for identifiers. These conditions are the same as for the <i>parameter_name</i> parameter of the create procedure command. • <i>SQL_statements</i> – the actions the user-defined Web service is to take. Any number and kind of SQL statements can be included, with the exception of create view, create default, create rule, create procedure, create trigger, and use. These conditions are the same as for the <i>SQL_statements</i> parameter of the create procedure command. • type – can be soap, raw, or xml: <ul style="list-style-type: none"> • soap implies an HTTP POST request and must be compliant with all the SOAP rules. The data is returned in SQL/XML format. • raw indicates that the output is to be sent without any alteration or reformatting. This implies an HTTP GET request. The invoked stored procedure can specify the exact output.

- xml indicates that the result set output is returned in SQL/XML format. This implies an HTTP GET request.

Note For datatype mappings between ASE stored procedures and SOAP user-defined Web services, see “ASE-to-SOAP datatype mappings for the create service command” in Appendix C, “SOAP and Adaptive Server Enterprise Datatype Mapping.”

Example 1

In this example, a user-defined Web service, *rawservice*, of type *raw* is created to return the version of the current database. The *create service* command is entered from the *isql* command line for the *pubs2* database:

```
1> use pubs2
2> go
1> create service rawservice type raw as select
'<html><h1>' + @@version + '</h1></html>'
2> go
```

The newly created user-defined Web service must then be deployed:

```
1> sp_webservices 'deploy', 'all'
2> go
```

For example, if the sample file is named *testraw.html* and it is copied to *\$\$SYBASE/WS-15_0/producer* (*%SYBASE%\WS-15_0\producer* on Windows), you can access the page at <https://myhost:8182/testraw.html>, where the *username*, *password*, and *database* are *bob*, *bob123*, and *pubs2*, respectively. Click “Access *rawservice*” to display the result.

Note For details on the *deploy* option for *sp_webservices*, see “Using *sp_webservices* with user-defined Web services” on page 51.

The WSDL for the newly created user-defined Web service can be found at the following URL:

```
https://myhost:8182/services/pubs2?wsdl
```

The output, an Adaptive Server Enterprise version string, is displayed in an HTML `<h1>` tag in the browser window:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
    <title>Inovke version</title>
    <link rel="stylesheet" type="text/css" href="ws.css">
```

```

</head>
<body>
  <form method="POST" action=services>
    <p>Username: <input type="text" name="username"</p>
    <p>Password: <input type="text" name="password"</p>
    <p>Database: <input type="text" name="dboralias"</p>
    <p><input type="hidden" value="rawservice" name="method"</p>
    <p><input type="submit" value="Access rawservice" name="B2">
  </form>
</body>
</html>

```

Example 2

In this example, a user-defined Web service, `xmlservice`, of type `xml` is created to return the version of the current database. The create service command is entered from the `isql` command line for the `pubs2` database:

```

1> use pubs2
2> go
1> create service xmlservice userpath "testing" type xml
as select @@version
2> go

```

The newly created user-defined Web service must then be deployed:

```

1> sp_webservices 'deploy', 'xmlservice'
2> go

```

Note For details on the deploy option for `sp_webservices`, see “Using `sp_webservices` with user-defined Web services” on page 51.

The WSDL for user-defined Web service can be found at the following URL:

```
https://localhost:8182/services/pubs2/testing?wsdl
```

For example, if the HTML page is named `testxml.html`, and you copy the HTML file to `$$SYBASE/WS-15_0/producer (%SYBASE%\WS-15_0\producer` on Windows). Access the page at `https://myhost:8182/testxml.html` and input these parameters:

- bob – the user ID
- bob123 – the password
- pubs2/testing – the database

Click “Retrieve Version” to display the result.

```

<html>
  <head>

```

```
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Inovke version</title>
<link rel="stylesheet" type="text/css" href="ws.css">
</head>
<body>
  <form method="POST" action=services>
    <p>Username: <input type="text" name="username"</p>
    <p>Password: <input type="text" name="password"</p>
    <p>Database: <input type="text" name="dboralias"</p>
    <p><input type="hidden" value="xmlservice" name="method"</p>
    <p><input type="submit" value="Retrieve Version" name="B1">
  </form>
</body>
</html>
```

Example 3

In this example, a user-defined Web service is made available to a SOAP client to execute the stored procedure `sp_who`. One argument is supplied, and the optional `userpath` token is specified:

```
create service sp_who_service userpath
'myservices/args' type soap @loginname varchar(30) as
exec sp_who @loginname
```

The Web service is created as `sp_who_service` in the `pubs2` database and, after being deployed, it is accessible at the following URL:

```
http://localhost:8181/pubs2/myservices/args/sp_who_service
```

The WSDL for the service is available at the following URL:

```
http://localhost:8181/pubs2/myservices/args?wsdl
```

The signature for the Web method, described in the WSDL file, is as follows:

```
DataReturn[] sp_who_service (xsd:string username,
xsd:string password, xsd:string loginname)
```

The new service is invoked by a SOAP client with one parameter, `loginname`, of type `varchar(30)`.

drop service

The drop service command removes a user-defined Web service from the current database. Both the metadata and the corresponding stored procedure are removed.

Note You must undeploy a user-defined Web service before you can drop it. For details on the undeploy option for `sp_webservices`, see “Using `sp_webservices` with user-defined Web services” on page 51.

Syntax	<code>drop service service-name</code>
Parameters	<ul style="list-style-type: none"> • <i>service-name</i> – the name for the user-defined Web service. This name can be any name that is valid for a stored procedure. If you specify the name of a service that does not exist, an exception results. Also, you cannot drop a service that is currently in use by another session.
Example	<p>This example drops the user-defined Web service named <code>sp_who_service</code>:</p> <pre>drop service sp_who_service</pre>

Using `sp_webservices` with user-defined Web services

The following `sp_webservices` options are used for user-defined Web services:

- `addalias` – creates a database alias.
- `deploy` – deploys a user-defined Web service.
- `dropalias` – drops a database alias.
- `listudws` – lists the proxy tables mapped to a WSDL file.
- `listalias` – lists a database alias or aliases.
- `undeploy` – undeploys a user-defined Web service.

addalias

The add alias option is used to create an alias representing a database name.

Syntax	<code>sp_webservices 'addalias' alias_name , database_name</code>
Parameters	<ul style="list-style-type: none"> • <i>alias_name</i> The alias for the specified database. This parameter is required. • <i>database_name</i>

The name of the database for which the alias is specified. This parameter is required.

Usage

An alias provides greater control in specifying the portion of the URL representing the database name. Used with the `userpath` option of the `create service` command, an alias provides complete control over the URL used to access a user-defined Web service.

deploy

The `deploy` option is used to deploy a user-defined Web service, making it accessible to the ASE Web Services Engine through HTTPS.

Note Only HTTP POST requests are supported to access deployed services because these commands embed all arguments within the URL for GET HTTP requests, which cannot be encrypted. You must enable HTTPS to encrypt the HTTP POST request.

Syntax

```
sp_webservices 'deploy', ['all' | 'service_name']
```

Parameters

- `all`
Specifies that all user-defined Web services are to be deployed for the current database.
- `service_name`

The name of the user-defined Web service to be deployed.

Usage

The `deploy` and `undeploy` options are used to control when user-defined Web services are available. The `webservices_role` privilege is required for this option.

If the `all` parameter is specified, the ASE Web Services Engine deletes its internal cache of user-defined Web services and rereads all metadata about user-defined Web services from Adaptive Server Enterprise.

Note You cannot drop or rename a user-defined Web service that is currently deployed.

dropalias

The `dropalias` option is used to drop an alias representing a database name.

Syntax	<code>sp_webservices 'dropalias' <i>alias_name</i></code>
Parameters	<ul style="list-style-type: none">• <i>alias_name</i> The alias to be dropped.
Usage	An alias cannot be dropped if it is being referenced by a deployed user-defined Web service. To drop the alias, you must first undeploy the user-defined Web service that references the alias.

listudws

The `listudws` option is used to list user-defined Web services for the current database.

Syntax	<code>sp_webservices 'listudws' [, '<i>service_name</i>']</code>
Parameters	<ul style="list-style-type: none">• <i>service_name</i> The name of the user-defined Web service to be listed.
Usage	If the <i>service_name</i> parameter is not specified, all user-defined Web services are listed.

listalias

The `listalias` option is used to list all aliases.

Syntax	<code>sp_webservices 'listalias'</code>
Usage	All aliases are listed.

undeploy

The `undeploy` option is used to make a user-defined Web service inaccessible to the ASE Web Services Engine through HTTPS.

Syntax	<code>sp_webservices 'undeploy', ['all' '<i>service_name</i>']</code>
Parameters	<ul style="list-style-type: none">• <code>all</code> Specifies that all user-defined Web services are to be undeployed for the current database.• <i>service_name</i> The name of the user-defined Web service to be undeployed.

Usage The deploy and undeploy options are used to control when user-defined Web services are available. The `webservices_role` privilege is required for this option.

Security for user-defined Web services

The system role `webservices_role` has been added to Adaptive Server Enterprise for the Web Services feature. This role is required to use the deploy and undeploy options for `sp_webservices`. To execute a user-defined Web service, a valid login and permissions to execute the corresponding stored procedure are required.

To create, drop, and execute user-defined Web services, you need the same privileges as are necessary to create, drop, and execute stored procedures in Adaptive Server Enterprise. See the Adaptive Server Enterprise *Security Administration Guide* for details on how to set the proper privileges using the `grant` and `revoke` commands.

Note For the most current information on security in ASE Web Services, see the Adaptive Server Enterprise *Release Bulletin*.

Auditing for user-defined Web services

User-defined Web services are modeled as stored procedures within Adaptive Server Enterprise. In manipulating user-defined Web services, Adaptive Server Enterprise generates the following events using the existing auditing coverage for stored procedures:

- The creation of a user-defined Web service – Event 11 named "Create Procedure" is generated
- The dropping of a user-defined Web service – Event 28 named "Drop Procedure" is generated
- The execution of a user-defined Web service – Event 38 named "Execution of Stored Procedure" is generated

For detailed information on existing auditing functionality, see the Adaptive Server Enterprise *Security Administration Guide*.

In addition to existing auditing functionality, Adaptive Server Enterprise provides two audit events for the deploy and undeploy options of `sp_webservices`.

Audit records are stored in the `sysaudits` system table. You can enable auditing for Web services with the following command:

```
sp_audit "security", "all", "all", "on"
```

Auditing `sp_webservices` 'deploy'

Audit event number 110 corresponds to the `deploy` option of `sp_webservices`.

Example 1

This example shows an audit table entry for the following command entered in the `pubs2` database by the user `bob`:

```
sp_webservices 'deploy', 'all'
```

The corresponding audit table entry lists 110, `bob`, and `pubs2` as values in the `event`, `loginname`, and `dbname` columns, respectively. The `extrainfo` column contains the following:

```
webservices_role; deploy_all; ; ; ; bob/ase;
```

Example 2

This example shows an audit table entry for the following command entered in the `pubs2` database by the user `bob`:

```
sp_webservices 'deploy', 'rawservice'
```

The corresponding audit table entry lists 110, `bob`, and `pubs2` as values in the `event`, `loginname`, and `dbname` columns, respectively. The `extrainfo` column contains the following:

```
webservices_role; deploy; ; ; ; bob/ase;
```

For a full description of `sysaudits` table columns, see the Adaptive Server Enterprise *Security Administration Guide*.

Auditing `sp_webservices` 'undeploy'

Audit event number 111 corresponds to the `undeploy` option of `sp_webservices`.

Example 1

This example shows an audit table entry for the following command entered in the `pubs2` database by the user `bob`:

```
sp_webservices 'undeploy', 'all'
```

The corresponding audit table entry lists 111, bob, and pubs2 as values in the event, loginname, and dbname columns, respectively. The extrainfo column contains the following:

```
webservices_role; undeploy_all; ; ; ; bob/ase;
```

Example 2

This example shows an audit table entry for the following command entered in the pubs2 database by the user bob:

```
sp_webservices 'undeploy', 'rawservice'
```

The corresponding audit table entry lists 111, bob, and pubs2 as values in the event, loginname, and dbname columns, respectively. The extrainfo column contains the following:

```
webservices_role; deploy; ; ; ; bob/ase;
```

For a full description of sysaudits table columns, see the Adaptive Server Enterprise *Security Administration Guide*.

ASE Web Services logging

By default, ASE Web Services logs only informational and error messages. For details on how to log more detailed information, contact Sybase Technical Support.

This section concerns the following topics:

- ASE Web Services log files
- Rolling over log files

ASE Web Services log files

ASE Web Services logs activity in these logs:

- *webservice.log* – contains information and error messages for the Web Services producer and consumer.
- *http.log* – contains all HTTPS requests in the NCSA Request Log format. An HTTPS request exists for each Web method invoked

Rolling over log files

Logging is implemented in ASE Web Services using the Apache log4j framework. For information on specific log4j parameters and implementing a rollover policy, refer to the following Web documentation:
<http://jakarta.apache.org/log4j/docs/>.

Using Sybase Central

The Sybase Central plug-in for ASE Web Services enables you to perform the following tasks:

- Configure Web Services from Sybase Central
- Consume a Web service
- Execute a Web service
- Delete a Web method
- Add a user-defined Web service
- Add an alias
- Display properties for a Web method

For details, refer to the online help for ASE Web Services in Sybase Central.

This chapter discusses the following:

Topic	Page
Apache sample client	59
Microsoft .NET sample client	64

Tools are provided under the *samples* directory to create and run sample clients. This chapter documents sample applications provided for the ASE Web Services Engine.

Apache sample client

This section describes the sample client and script found in the `$SYBASE/WS-15_0/samples/apacheclient` directory in UNIX, or in the `%SYBASE%\WS-15_0\samples\apacheclient` directory in Windows.

Note If you intend to run the Apache sample client on a machine other than the one on which ASE Web Services is installed, you must copy the contents of the `/apacheclient/lib` directory to that machine.

Creating the sample client

To use the sample script provided, you must first create the sample client.

❖ Creating the sample client

- Make sure the JRE variable points to your JRE by changing, if necessary, the variable definitions in all scripts in the *apacheclient* directory. You must use JRE version 1.4.2 or later. By default, the JRE supplied in the UNIX `$SYBASE_JRE` or Windows `%SYBASE_JRE%` directory is used.

Once you have created an ASE Web Services client, you can run the sample script to execute stored procedures and SQL statements. This script can be found in the *apacheclient* directory.

Using runexecute

The *runexecute* script executes a stored procedure or Transact-SQL statement on Adaptive Server Enterprise through ASE Web Services. This sample application invokes the *execute* Web method.

Syntax `runexecute "web_service_URL" aseServerName user_ID password
 "SQLX_option" output_class count "sql_statement"`

Parameters

- *web_service_URL*
The location of the Web service being used.
- *aseServerName*
SOAP string indicating the name of the Adaptive Server Enterprise server in the *interfaces* file or LDAP server.
- *user_ID*
The user ID needed to log in to the Adaptive Server Enterprise.
- *password*
The password needed to log in to the Adaptive Server Enterprise.
- *SQLX_option*
String indicating one or more option parameters. These parameters specify characteristics of the SQLX result set. The following are valid option parameters:
 - `binary={hex | base64}`
 - `columnstyle={element | attribute}`
 - `format={yes | no}`
 - `header={yes | no}`
 - `nullstyle={attribute | omit}`
 - `prefix="value"`
 - `root={yes | no}`
 - `rowname="value"`

- `schemaloc="value"`
- `statement={yes | no}`
- `tablename="value"`
- `targetns="value"`

You must provide values for *value*. For more information on SQLX functions and options, see *XML Services in Adaptive Server Enterprise*.

- *output_class*

The kind of output desired. The following are valid values for this parameter:

- `schema` – returns an XML schema.
- `dtd` – returns an XML DTD.
- `data` – returns a result set.
- `all` – returns schema, DTD, and data.

- *count*

The number of times to execute the statement. If the value of `count` is greater than 1, a session is created, and a persistent connection is used.

- *sql_statement*

The statement to be executed on Adaptive Server Enterprise. This statement must be delimited by double quotes.

Example 1

This example checks the version number for Adaptive Server Enterprise using a `select` statement.

```
runexecute "http://johndoe-sun:8183/services/ase"
johndoe-sun sa nopasswordspecified "tablename=ws" all 1
"select @@version"
```

ASE Web Services returns an XML schema, a DTD, and a result set containing the result of the executed statement.

Example 2

This example executes a stored procedure called `booksales` on the `pubs2` database. The stored procedure returns the number of copies sold for a specified book title ID.

```
runexecute "http://johndoe-sun:8183/services/ase"
johndoe-sun sa nopasswordspecified
"columnstyle=attribute,format=no,rowname=wsrow,prefix=
Unnamedcol,nullstyle=attribute,header=yes" all 1
"execute booksales MC2222"
```

ASE Web Services returns an XML schema, a DTD, and a result set containing the result of the executed statement.

This is the result set returned:

```
<?xml version="1.0" ?>
<resultset
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<wsrow title="Silicon Valley Gastronomic Treats"
total_sales="2032" Unnamedcol1="Books sold"/>
</resultset>
```

This is the DTD returned:

```
<!DOCTYPE ws [
<!ELEMENT resultset (row*)>
<!ELEMENT row (title, total_sales, Unnamedcol1)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT total_sales (#PCDATA)>
<!ELEMENT Unnamedcol1 (#PCDATA)>
]>
```

This is the schema returned:

```
<?xml version="1.0" ?>
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:sqlxml="http://www.iso-standards.org/mra/9075/
sqlx">
<xsd:import
namespace="http://www.w3.org/2001/XMLSchema"
schemaLocation="http://www.iso-standards.org/mra/
9075/sqlx.xsd" />
<xsd:complexType
name="RowType.resultset">
<xsd:attribute name="title"
type="VARCHAR_80" use="required"/>
<xsd:attribute name="total_sales" type="INTEGER"
use="optional"/>
```

```

<xsd:attribute name="Unnamedcol1"
  type="VARCHAR_24" use="optional"/>
</xsd:complexType>
<xsd:complexType
  name="TableType.resultset">
  <xsd:sequence>
    <xsd:element name="wsrow"
      type="RowType.resultset" minOccurs="0"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="VARCHAR_80">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="80"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="INTEGER">
  <xsd:restriction base="xsd:integer">
    <xsd:maxInclusive value="2147483647"/>
    <xsd:minInclusive value="-2147483648"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="VARCHAR_24">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="24"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:element name="resultset"
  type="TableType.resultset"/>
</xsd:schema>

```

Example 3

This example executes a SQL query on the pubs2 database. The query returns the last names and cities of residence for authors who do not live in the same city as their publisher.

```

runexecute "http://johndoe-sun:8183/services/ase"
johndoe-sun sa nopasswordspecified
"tablename=ws,header=yes,schemaloc='http://www-
edm/remote/avr/xmltestdir/resultset.xsd',targetns='htt
p://www-edm/remote/avr/xmltestdir/'" data 1 "select
distinct au_lname, authors.city from publishers,
authors where authors.city not in (select city from
publishers where authors.city = publishers.city)"

```

ASE Web Services returns a result set containing the result of the executed statement.

Microsoft .NET sample client

This section describes the sample client and script found in the `$$SYBASE/WS-15_0/samples/ms.net/Execute/bin/Release` directory in UNIX, or in the `%%SYBASE%%\WS-15_0\samples\ms.net\Execute\bin\Release` directory in Windows.

Downloads for Microsoft .NET can be found at the following URL:

<http://msdn.microsoft.com/library/default.asp?url=/downloads/list/netdevframework.asp>

Note This URL is current as of the date of publication for this document but may change over time.

Creating the sample client

To use the sample script provided, you must first create the sample client. Once you have created the sample client, you can run the sample script documented in this section. This script can be found in the *Release* directory.

Using Execute.exe

Execute.exe executes a stored procedure or Transact-SQL statement on Adaptive Server Enterprise through ASE Web Services. This sample application invokes the *execute* Web method.

Syntax *Execute.exe* "*web_service_URL*" *aseServerName* *user_ID* *password*
 "*SQLX_option*" *output_class* *count* "*sql_statement*"

- Parameters
- *web_service_URL*
The location of the Web service being used.
 - *aseServerName*
SOAP string indicating the name of the Adaptive Server Enterprise server in the *sql.ini* file or LDAP server.
 - *user_ID*
The user ID needed to log in to the Adaptive Server Enterprise.
 - *password*

The password needed to log in to the Adaptive Server Enterprise.

- *SQLX_option*

String indicating one or more option parameters. These parameters specify characteristics of the SQLX result set. The following are valid option parameters:

- `binary={hex | base64}`
- `columnstyle={element | attribute}`
- `format={yes | no}`
- `header={yes | no}`
- `nullstyle={attribute | omit}`
- `prefix="value"`
- `root={yes | no}`
- `rowname="value"`
- `schemaloc="value"`
- `statement={yes | no}`
- `tablename="value"`
- `targetns="value"`

You must provide values for *value*. For more information on SQLX functions and options, see *XML Services in Adaptive Server Enterprise*.

- *output_class*

The kind of output desired. The following are valid values for this parameter:

- `schema` - returns an XML schema.
- `dtd` - returns an XML DTD.
- `data` - returns a result set.
- `all` - returns schema, DTD, and data.

- *count*

The number of times to execute.

- *sql_statement*

The statement to be executed on Adaptive Server Enterprise. This statement must be delimited by double quotes.

Example 1

This example checks the version number for Adaptive Server Enterprise.

```
Execute.exe "http://johndoe-sun:8183/services/ase"  
johndoe-sun sa nopasswordspecified "tablename=ws" all 1  
"select @@version"
```

ASE Web Services returns an XML schema, a DTD, and a result set containing the result of the executed statement.

Example 2

This example executes a stored procedure called booksales on the pubs2 database. The stored procedure returns the number of copies sold for a specified book title ID.

```
Execute.exe "http://johndoe-sun:8183/services/ase"  
johndoe-sun sa nopasswordspecified  
"columnstyle=attribute,format=no,rowname=wsrow,prefix=  
Unnamedcol,nullstyle=attribute,header=yes" all 1  
"execute booksales MC2222"
```

ASE Web Services returns an XML schema, a DTD, and a result set containing the result of the executed statement.

Example 3

This example executes a SQL query on the pubs2 database. The query returns the last names and cities of residence for authors who do not live in the same city as their publisher.

```
Execute.exe "http://johndoe-sun:8183/services/ase"  
johndoe-sun sa nopasswordspecified  
"tablename=ws,header=yes,schemaloc='http://www-  
edm/remote/ivr/xmltestdir/resultset.xsd',targetns='htt  
p://www-edm/remote/ivr/xmltestdir/'" data 1 "select  
distinct au_lname, authors.city from publishers,  
authors where authors.city not in (select city from  
publishers where authors.city = publishers.city)"
```

ASE Web Services returns a result set containing the result of the executed statement.

Troubleshooting

This section describes troubleshooting procedures and error messages.

This chapter discusses the following:

Topic	Page
Troubleshooting issues	69
Diagnostic tools	74
Messages	76

Troubleshooting issues

The following are issues that can assist you in troubleshooting ASE Web Services.

Remote server class definition setting

Issue The `sp_webservices add` command may return the following error when generating proxy tables:

```
Warning: Row size (3347 bytes) could exceed row size limit, which is 1962
bytes.
Msg 208, Level 16, State 1:
Server 'JMALVARADO', Line 1:
tempdb..ws_4338e6e122cd4ef0a not found. Specify owner.objectname or uses to
check whether the object exists (sp_help may produce lots of output).
No proxy tables were created for the WSDL URL:
[http://www.xignite.com/xquotes.asmx?WSDL]
(return status = 0)
```

This error occurs because the remote server representing the ASE Web Services Engine has been added using `sp_addserver` with a class other than “sds.” To verify that this is so, use `sp_helpserver` in `isql`:

```
sp_helpserver ws
```

Here, `ws` is the name of the ASE Web Services Engine. This is the default. The remote server class is returned in the indicated column of the result:

```
name network_name class ...
-----
ws ws null ...
```

User action Change the class of the remote server to “sds” by using `sp_dropserver` and `sp_addserver` in isql:

```
sp_dropserver ws_name
...
sp_addserver ws_name, sds, ws_name
```

Here, `ws_name` is the name chosen for the ASE Web Services Engine.

Unmapped RPC/encoded Web method

Issue If an RPC/encoded Web method has no input or output parameters, it cannot be mapped to a proxy table. A proxy table for a Web method without parameters would have no columns. A table with no columns cannot be created in Adaptive Server Enterprise.

User action Modify the Web method to include an input or output parameter.

Issue Only simple types are mapped to columns. Complex types or arrays used in RPC/encoded Web methods result in the Web method not being mapped to a proxy table.

User action Modify the Web method to use only simple types and arrays.

Truncated document/literal results

Issue If a Web service returns more data than the value of the `@@textsize` global variable, the data is truncated to the size specified by `@@textsize`. Consequently, the data returned may not form a valid XML document.

User action No error is raised because this is the expected behavior for Adaptive Server Enterprise when text or image data is truncated with a `select` command. However, a warning is logged for the ASE Web Services Engine, so check your `consumer.log` file.

Starting ASE Web Services Engine

Issue The `runws` script does not successfully start a ASE Web Services Engine.

- User action
- 1 Make sure the port you are attempting to use is not already in use by another process.
 - 2 Make sure you have the correct JRE installed. ASE Web Services requires JRE 1.4.2 or later.
To check your JRE version, enter the following at your command prompt:

```
java -version
```
 - 3 If you want Web Services to run with a properties file other than *ws.properties*, you must specify the absolute path for the file. For example, to run the ASE Web Services Engine with a different properties file:

```
C:\sybase\WS-15_0\bin\runws -f  
C:\sybase\WS-15_0\props\myfile.properties
```
- Issue
- The ASE Web Services Engine finds the specified *ase_service_name* in the *interfaces* file, but the *producer.log* shows the following error messages:
- ```
INFO [main] - Error locating libtcl.cfg file.
INFO [main] - java.io.FileNotFoundException: LDAP
config File does not exist
```

## Locating WSDL

- Issue
- A client connecting to the ASE Web Services Engine through a Web browser cannot find the WSDL file, or the ASE Web Services Engine cannot find the WSDL file to perform proxy-table mapping.
- User action
- Verify that the ASE Web Services Engine is running. If you are using the ASE Web Services Engine through a browser, make sure the browser URL indicates `https://` for an SSL connection and `http://` for a standard connection.

## Specifying entries in ws.properties

- Issue
- Because the backslash (\) symbol is used as an escape character, entries that use single backslash symbols are not interpreted correctly. For example:
- ```
com.sybase.ase.ws.interfaces = d:\sybase\ini\sql.ini
```
- User action
- Escape the backslash with another backslash:
- ```
com.sybase.ase.ws.interfaces = d:\\sybase\\ini\\sql.ini
```
- You can also use forward slashes:

```
com.sybase.ase.ws.interfaces = d:/sybase/ini/sql.ini
```

## Windows NT command-line arguments

**Issue** Scripts do not run on Windows NT when no space is placed between arguments and argument values. For example, the following invocation of the *configssl* script will not execute the script:

```
configssl -dhostname
```

**User action** Place a space between an argument and its value:

```
configssl -d hostname
```

## Run or stop scripts fail

**Issue** The *runws* or *stopws* script fails to execute.

**User action** If either of these scripts fail to execute, do the following:

- Verify that your Adaptive Server Enterprise is running.
- Make sure that the user name and password you specified are valid to log in to your Adaptive Server Enterprise.
- Check the *producer.log* or *consumer.log* file for any error messages.
- Verify that the *ase\_service\_name* provided can be found on an LDAP server pointed to by the *libtcl.cfg* file or in the *interfaces* file for Adaptive Server Enterprise.

---

**Note** On Windows systems, the *interfaces* file is named *sql.ini*.

---

- Verify that the user name has *sa\_role* privileges.

## Null passwords

**Issue** The password for an Adaptive Server Enterprise user may be set to a null string.

**User action** Use the token *nopasswordspecified* anywhere the password is required, including the *runws* and *stopws* scripts.

## Specifying SOAP endpoints with SSL

|             |                                                                                                                                                                                                                                                                                                                                                   |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Issue       | ASE Web Services methods or sample applications do not return results with the <i>aseServerName</i> specified at invocation.                                                                                                                                                                                                                      |
| User action | Make sure the <i>aseServerName</i> name is a valid SOAP endpoint. If you are using a DNS alias, make sure the alias resolves to a valid SOAP endpoint. If you are using SSL, make sure the endpoint specified by <i>aseServerName</i> is the same name you supplied in creating an SSL certificate with the <i>configssl</i> script. For example: |

```
configssl -d mydomainhostname -h 8182
```

Here, the value of *aseServerName* supplied when invoking an ASE Web Services method or sample application must be “https://mydomainhostname:8182”. The method or sample application will not return results if you substitute “localhost” or an IP address for *aseServerName* when using SSL.

## Abnormal termination of sp\_webservices ‘add’

|             |                                                                                                                                                                                                     |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Issue       | Proxy tables created during execution of the sp_webservices ‘add’ option remain after an abnormal termination of sp_webservices, as with a Ctrl+C interrupt or an Adaptive Server Enterprise crash. |
| User action | Use the remove option to delete any proxy tables created by the add option when sp_webservices terminated abnormally.                                                                               |

## Web Services proxy table restrictions

|             |                                                                                                                                                                                                                                                                                                                 |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Issue       | A proxy table with the same name as a Web Services proxy table generated through execution of the add option of sp_webservices may be erroneously designated as a Web Services proxy table. Joins are processed for Web Services proxy tables differently than joins for regular proxy tables.                  |
| User action | Confine Web Services proxy tables to a unique database.                                                                                                                                                                                                                                                         |
| Issue       | The reserved word or is not supported in the where clause for a query involving Web Services proxy tables. This may result in errors for queries that are translated internally into queries that use the reserved word or. For example, the following query selects from the Web Services proxy table testint: |

```
select * from testint where _in0 in (1, 2)
```

This query is translated internally into the following:

```
select * from testint where _in0 = 1 or _in0 = 2
```

Because the internal translation uses the reserved word `or`, the user-submitted query results in an error.

User action

Alter the query so that its internal translation does not use the reserved word `or`. For example, the user-submitted query just described can be altered with the use of temporary tables:

```
create table a (col int)
insert into a values (1)
insert into a values (2)
select * from testint where _in0 in (select col from a)
```

## sysattributes table entry

Issue

The following error occurs when `sp_webservices 'add'` is executed:

```
Msg 5629, Level 16, State 1:
Line 1:
Cannot start a remote distributed transaction
participant as the local server is not named. Please
contact a user with System Administrator role.
```

User action

You must have an entry for your Adaptive Server Enterprise in the `sysattributes` table. To provide an entry, use the `sp_addserver` stored procedure:

```
sp_addserver ase_entry, local, ase_entry
```

where `ase_entry` is the local name for your Adaptive Server Enterprise from your `interfaces` or `sql.ini` file. Then, restart Adaptive Server Enterprise.

## Diagnostic tools

ASE Web Services provides detailed logging for diagnostics and JDBC-level tracing to help identify connectivity problems.



## Detailed logging

To enable detailed logging for diagnostics, modify the *logging.properties* file in the *props* directory as appropriate, and then restart the ASE Web Services Engine.

The following is an example of the content of the *logging.properties* file as it is delivered with ASE Web Services:

```
Set logging level for ASE Web Services Consumer
log4j.logger.com.sybase.ase.ws.sds=INFO, C
#log4j.logger.com.sybase.ase.ws.sds=DEBUG, C
log4j.additivity.com.sybase.ase.ws.sds=false

Set logging level for ASE Web Services Producer
log4j.logger.com.sybase.ase.ws.producer=INFO, P
#log4j.logger.com.sybase.ase.ws.producer=TRACE, P
log4j.additivity.com.sybase.ase.ws.producer=false
```

For detailed, diagnostic logging, modify *logging.properties* as follows:

```
Set logging level for ASE Web Services Consumer
#log4j.logger.com.sybase.ase.ws.sds=INFO, C
log4j.logger.com.sybase.ase.ws.sds=DEBUG, C
log4j.additivity.com.sybase.ase.ws.sds=false

Set logging level for ASE Web Services Producer
#log4j.logger.com.sybase.ase.ws.producer=INFO, P
log4j.logger.com.sybase.ase.ws.producer=TRACE, P
log4j.additivity.com.sybase.ase.ws.producer=false
```

## Enabling JDBC-level tracing

To enable JDBC-level tracing, refer to the appropriate jConnect documentation.

## Messages

Table 6-1 lists messages for ASE Web Services.

**Table 6-1: ASE Web Services messages**

| Message number | Message text                                                                                                                       |
|----------------|------------------------------------------------------------------------------------------------------------------------------------|
| 15200          | No web methods mapped to proxy tables for the WSDL URI [%s].                                                                       |
| 15201          | WSDL URI in most cases has the suffix ?WSDL. Please verify WSDL URI.                                                               |
| 15202          | Web Method [%s] not mapped to proxy table because of unsupported datatype.                                                         |
| 15203          | Received request to execute an unknown procedure [%s].                                                                             |
| 15204          | Caught IOException. This usually indicates an error in communications between ASE and the ASE Web Services Engine.\nDetails: [%s]. |
| 15205          | Caught SQLException. This usually indicates an error retrieving meta data from ASE.\nDetails: [%s].                                |
| 15206          | Caught an Unknown Exception: Details: [%s].                                                                                        |
| 15207          | Caught Remote Web Method Exception (AxisFault). This indicates an error in the remote web method.\nDetails: [%s].                  |
| 15208          | Caught Mapping Exception. This indicates an error in mapping the web method arguments to ASE types.\n Details: [%s].               |
| 15209          | Caught Service Exception. This usually indicates an incorrect WSDL file.\n Details: [%s].                                          |
| 15210          | Received XML input to the webmethod that was is not well formed.                                                                   |
| 15211          | Error in invoking web method (MalformedURLException) Details: [%s].                                                                |
| 15212          | Caught RemoteException. This usually indicates an error in the network transmission.\n Details: [%s].                              |
| 15213          | Error in invoking web method (Unknown Exception): Details [%s].                                                                    |
| 15214          | Aborting invocation of web method [%s] from proxy table [%s] because the web method expects [%s] arguments and [%s] were received. |
| 15215          | Parameter count/type mismatch. Check the number and types of the parameters passed to the built-in function, ws_admin.             |
| 15220          | A user-defined Web service by that name already exists.                                                                            |
| 15221          | Cannot drop alias when it is being referenced by a service.                                                                        |
| 15216          | Unknown operation, '%.*s', specified to built-in function ws_admin. Check parameter spelling and placement.                        |
| 15217          | Failure during update/insert/delete from sysattributes.                                                                            |
| 15218          | Cannot retrieve current database name.                                                                                             |

| <b>Message number</b> | <b>Message text</b>                                                                                                                                                                   |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15219                 | Cannot retrieve object ID for '%.*s'.                                                                                                                                                 |
| 19307                 | Generating proxy tables using sds [%1!] for WSDL URI: [%2!].                                                                                                                          |
| 19308                 | Found WSDL Match for [%1!].                                                                                                                                                           |
| 19309                 | The WSDL URI [%1!] was not found in the system.                                                                                                                                       |
| 19310                 | Updating timeout entries for WSDL URI [%1!] with [%2!].                                                                                                                               |
| 19311                 | Removing all web service meta data....                                                                                                                                                |
| 19312                 | Deleting entries for WSDL URI [%1!].                                                                                                                                                  |
| 19313                 | To remove a webservice, a valid SDS server must be supplied. The SDS server [%1!] was not found. Please use sp_addserver to add the SDS server.                                       |
| 19319                 | To add a webservice, a valid SDS server must be supplied. \n The SDS server [%1!] was not found. \nPlease use sp_addserver to add the SDS server.                                     |
| 19320                 | Verify that the ASE Web Services Engine is running.                                                                                                                                   |
| 19321                 | To add a webservice, a valid WSDL URI must be specified.                                                                                                                              |
| 19322                 | The WSDL URI [%1!] specified is already in the system. \n Please use sp_webservices remove first.                                                                                     |
| 19323                 | To list information about a webservice not loaded in the system, a SDS server must be supplied. The SDS server [%1!] was not found. \n Please use sp_addserver to add the SDS server. |
| 19324                 | Must specify a specific wsdl uri to modify.                                                                                                                                           |
| 19325                 | Must specify something to change for modify.                                                                                                                                          |
| 19326                 | [%1!] is not a valid option for sp_webservices modify.                                                                                                                                |
| 19327                 | Must specify item=value syntax for modify.                                                                                                                                            |
| 19408                 | Cannot drop alias because it does not exist.                                                                                                                                          |
| 19409                 | Specify the name of the alias to add.                                                                                                                                                 |
| 19410                 | Specify the database name associated with this alias.                                                                                                                                 |
| 19412                 | Cannot rename a deployed service.                                                                                                                                                     |

For help information for sp\_webservices, enter sp\_webservices help at the isql command line.



PART 5

# Appendixes

This section contains the appendixes for this book.

---

# Installation Contents

This appendix describes the contents of the ASE Web Services installation.

## ASE Web Services directory tree

ASE Web Services is installed at the same level as the root directory for Adaptive Server Enterprise. The ASE Web Services root directory is named *WS-15\_0* and consists of the following subdirectories:

**Table A-1: ASE Web Services directories**

| <b>Directory name</b> | <b>Contents</b>                                                                           |
|-----------------------|-------------------------------------------------------------------------------------------|
| <i>bin</i>            | Scripts for configuring and running ASE Web Services.                                     |
| <i>lib</i>            | Java libraries and packages used by ASE Web Services.                                     |
| <i>logs</i>           | Default location for log files.                                                           |
| <i>producer</i>       | Files and subdirectories for ASE Web Services Engine at runtime.                          |
| <i>props</i>          | Files for ASE Web Services properties.                                                    |
| <i>samples</i>        | Sample scripts for building and running a sample client with the ASE Web Services Engine. |

## **bin directory contents**

The following table describes the contents of the *bin* directory.

**Table A-2: bin directory contents**

| <b>File/Directory name</b> | <b>Function</b>                                                                    |
|----------------------------|------------------------------------------------------------------------------------|
| <i>configssl</i>           | Configures SSL.                                                                    |
| <i>getpass.exe</i>         | Used for Adaptive Server Enterprise login. This file is only available on Windows. |
| <i>installws</i>           | Installs sp_webservices stored procedure.                                          |
| <i>runtcpmon</i>           | Creates a monitor to trace SOAP messages.                                          |
| <i>runws</i>               | Starts the ASE Web Services Engine.                                                |
| <i>stopws</i>              | Stops the ASE Web Services Engine.                                                 |

---

**Note** The Windows versions of these files have a *.bat* suffix.

---

## **lib directory contents**

The following table describes the contents of the *lib* directory.

**Table A-3: lib directory contents**

| <b>File/Directory name</b>   | <b>Function</b>                              |
|------------------------------|----------------------------------------------|
| <i>axis.jar</i>              | Apache Axis file                             |
| <i>commons-discovery.jar</i> | Apache Axis file                             |
| <i>commons-logging.jar</i>   | Apache Axis file                             |
| <i>javax.servlet.jar</i>     | Servlet library                              |
| <i>jaxrpc.jar</i>            | Apache Axis file                             |
| <i>log4j-1.2.4.jar</i>       | log4j logger                                 |
| <i>mail.jar</i>              | Apache Axis file                             |
| <i>org.mortbay.jetty.jar</i> | HTTP server                                  |
| <i>sacj.jar</i>              | Apache Axis file                             |
| <i>sqlx.jar</i>              | SQLX file                                    |
| <i>tools.jar</i>             | Java tools (for example, the javac compiler) |
| <i>ws.jar</i>                | Code for ASE Web Services Engine             |
| <i>ws_debug.jar</i>          | Debugger Code for ASE Web Services Engine    |



| <b>File/Directory name</b> | <b>Function</b>  |
|----------------------------|------------------|
| <i>wSDL4j.jar</i>          | Apache Axis file |
| <i>xercesImpl.jar</i>      | Xerces parser    |
| <i>xmlParserAPIs.jar</i>   | Xerces parser    |

## ***logs* directory contents**

The following table describes the contents of the *logs* directory.

**Table A-4: *logs* directory contents**

| <b>File/Directory name</b> | <b>Function</b>                                                                  |
|----------------------------|----------------------------------------------------------------------------------|
| <i>webservices.log</i>     | contains information and error messages for Web Services producers and consumers |
| <i>http.log</i>            | Logs Web server activity.                                                        |

## ***producer* directory contents**

The following table describes the contents of the *producer* directory.

**Table A-5: *producer* directory contents**

| <b>File/Directory name</b> | <b>Function</b>                                                                                         |
|----------------------------|---------------------------------------------------------------------------------------------------------|
| <i>index.html</i>          | Home page for the Web services browser interface.                                                       |
| <i>keystore</i>            | Holds encryption keys.                                                                                  |
| <i>ui</i>                  | Directory structure for Web services browser interface.                                                 |
| <i>WEB-INF</i>             | Directory structure required for Jetty.                                                                 |
| <i>wscertificate.cer</i>   | Auto-generated certificate for SSL. This file is present only the <i>configssl</i> script has been run. |

## ***props* directory contents**

The following table describes the contents of the *props* directory.

**Table A-6: *props* directory contents**

| <b>Directory name</b>     | <b>Function</b>                                          |
|---------------------------|----------------------------------------------------------|
| <i>logging.properties</i> | Configuration file for log4j                             |
| <i>ws.properties</i>      | All configuration parameters for ASE Web Services Engine |
| <i>wsmsg.properties</i>   | Configuration file for messages                          |

## ***samples* directory contents**

The *samples* directory contains precompiled and source code for both Apache and Microsoft .NET. You can use source code in your own applications. The following table describes the contents of the *samples* directory.

**Table A-7: *samples* directory contents**

| <b>Directory name</b> | <b>Function</b>                                                                 |
|-----------------------|---------------------------------------------------------------------------------|
| <i>apacheclient</i>   | Directory containing sample scripts for compiling and running the sample client |
| <i>ms.net</i>         | Directory containing samples for .NET                                           |

# Configuration Properties

This appendix discusses the following:

| Topic                              | Page |
|------------------------------------|------|
| ws.properties                      | 85   |
| myres.properties                   | 87   |
| Specifying properties file entries | 89   |

## ws.properties

The *ws.properties* file contains the following configuration settings for ASE Web Services.

**Table B-1: ws.properties entries**

|                                                            |                                                                                                                                                                                                                                         |
|------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>com.sybase.ase.ws.producer.httpport</code>           | Indicates the port on which the ASE Web Services Engine should listen for an HTTP connection. The default entry is 8181.                                                                                                                |
| <code>com.sybase.ase.ws.consumer.cisport</code>            | Indicates the port on which the ASE Web Services Engine should listen for TDS. The default entry is 8183.                                                                                                                               |
| <code>com.sybase.ase.ws.logfilename</code>                 | Indicates where the log file for the ASE Web Services Engine should be placed. The default location for UNIX is <code>\$\$SYBASE/WS-15_0/logs/webservice.log</code> , or <code>%SYBASE%\WS-15_0\logs\webservice.log</code> for Windows. |
| <code>com.sybase.ase.ws.producer.jettylogfile</code>       | Indicates where the log file for HTTP requests should be placed. The default location is for UNIX is <code>\$\$SYBASE/WS-15_0/logs/http.log</code> , or <code>%SYBASE%\WS-15_0\logs\http.log</code> for Windows.                        |
| <code>com.sybase.ase.ws.producer.jettylogRetainDays</code> | Indicates the number of days the log is retained. The default is 90                                                                                                                                                                     |
| <code>com.sybase.ase.ws.producer.jettylogAppend</code>     | Indicates whether you can append to the log file. The default is true.                                                                                                                                                                  |

---

```
com.sybase.ase.ws.producer.jettylogExtended = false
```

Indicates whether to log more detailed information about external HTTP requests. The default is false.

---

```
com.sybase.ase.ws.producer.jettylogTimeZone = GMT
```

Indicates the time zone in which the log is created. The default is GMT.

---

```
com.sybase.ase.ws.interfaces
```

Indicates the location of the *interfaces* or *sql.ini* file for Adaptive Server Enterprise. The default location for UNIX is `$$SYBASE/interfaces`, or `%SYBASE%\ini\sql.ini` for Windows.

---

```
com.sybase.ase.ws.libtcl
```

Indicates the location of the *libtcl.cfg* file used to identify LDAP servers. The default location for 32-bit platforms for UNIX is `$$SYBASE/$$SYBASE_OCS/config/libtcl.cfg`, or `%SYBASE%\%SYBASE_OCS%\ini\libtcl.cfg` for Windows. The default location for 64-bit platforms for UNIX is `$$SYBASE/$$SYBASE_OCS/config/libtcl64.cfg`, or `%SYBASE%\%SYBASE_OCS%\ini\libtcl64.cfg` for Windows.

---

```
com.sybase.ase.ws.maxpostsize
```

Determines the maximum size of an incoming SOAP request. The default entry is 20000.

---

```
com.sybase.ase.ws.ui.activate
```

Determines whether the Web-based user interface is activated. The user interface is available at `https://hostname:https_port`. The default entry is true.

---

```
com.sybase.ase.ws.producer.tuning.maxthreads
```

Indicates the maximum number of threads in the thread pool servicing the HTTP port. The default entry is 250.

---

```
com.sybase.ase.ws.producer.tuning.minthreads
```

Indicates the minimum number of threads in the thread pool servicing the HTTP port. The default entry is 45.

---

```
com.sybase.ase.ws.producer.tuning.maxidletime
```

Indicates the maximum time in milliseconds a thread may remain idle. The default entry is 60000.

---

```
com.sybase.ase.ws.producer.tuning.ssl.maxthreads
```

Indicates the maximum number of threads in the thread pool servicing the HTTPS port. The default entry is 250.

---

```
com.sybase.ase.ws.producer.tuning.ssl.minthreads
```

Indicates the minimum number of threads in the thread pool servicing the HTTPS port. The default entry is 45.

---

```
com.sybase.ase.ws.producer.tuning.ssl.maxidletime
```

Indicates the maximum time in milliseconds a thread may remain idle. The default entry is 60000.

---

---

|                                                         |                                                                                                                                                                                                      |
|---------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>com.sybase.ase.ws.producer.ssl.keypassword</code> | Indicates the password for the SSL certificate. No default is provided.                                                                                                                              |
| <code>com.sybase.ase.ws.producer.ssl.keystore</code>    | Indicates the location of the keystore for SSL. The default location for UNIX is <code>\$\$SYBASE/WS-15_0/producer/keystore</code> , or <code>%SYBASE%\WS-15_0\producer\keystore</code> for Windows. |
| <code>com.sybase.ase.ws.producer.ssl.httpsport</code>   | Indicates the port on which the ASE Web Services Engine should listen for an HTTPS connection. The default entry is 8182.                                                                            |
| <code>com.sybase.ase.ws.producer.ssl.password</code>    | Indicates the keystore password for SSL. No default is provided.                                                                                                                                     |

---

## myres.properties

The *myres.properties* file is created when the Sybase Central plug-in performs configuration tasks for ASE Web Services. The *myres.properties* file contains the following configuration settings for ASE Web Services.

### **Table B-2: myres.properties entries**

---

|                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Web Services</i> – Set the following entries to configure Web Services. |                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>ws.ini</code>                                                        | Indicates the location of the <i>interfaces</i> or <i>sql.ini</i> file for Adaptive Server Enterprise. The default location for UNIX is <code>\$\$SYBASE/interfaces</code> , or <code>%SYBASE%\ini\sql.ini</code> for Windows.                                                                                                                                                                                                                            |
| <code>ws.libtcl</code>                                                     | Indicates the location of the <i>libtcl.cfg</i> file used to identify LDAP servers. The default location for 32-bit platforms for UNIX is <code>\$\$SYBASE/\$SYBASE_OCS/config/libtcl.cfg</code> , or <code>%SYBASE%\%SYBASE_OCS%\ini\libtcl.cfg</code> for Windows. The default location for 64-bit platforms for UNIX is <code>\$\$SYBASE/\$SYBASE_OCS/config/libtcl64.cfg</code> , or <code>%SYBASE%\%SYBASE_OCS%\ini\libtcl64.cfg</code> for Windows. |
| <code>ws.producer.port</code>                                              | Indicates the port for the ASE Web Services Engine.                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>ws.producer.log</code>                                               | Indicates the location of the <i>producer.log</i> file.                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>ws.producer.jettylogfile</code>                                      | Indicates where the log file for HTTP requests should be placed. The default location is for UNIX is <code>\$\$SYBASE/WS-15_0/logs/http.log</code> , or <code>%SYBASE%\WS-15_0\logs\http.log</code> for Windows.                                                                                                                                                                                                                                          |

---

---

SSL – Set the following entries to configure SSL.

---

ws.ssl.host

Indicates the name of the SSL host to be accessed.

---

ws.ssl.keystorelocation

Indicates the location of the keystore for SSL. The default location for UNIX is `$SYBASE/WS-15_0/producer/keystore`, or `%SYBASE%\WS-15_0\producer\keystore` for Windows.

---

ws.ssl.certificatepassword

Indicates the password for the SSL certificate. No default is provided.

---

ws.ssl.keystorepassword

Indicates the keystore password for SSL. No default is provided.

---

ws.consumer.name

Indicates the name of the ASE Web Services Engine as specified in the *interfaces* or *sql.ini* file.

---

ws.consumer.host

Indicates the host machine of the ASE Web Services Engine.

---

ws.consumer.port

Indicates the port number for the ASE Web Services Engine process.

---

ws.consumer.log

Indicates the location of the *consumer.log* file.

---

*installws* – Set the following entries to run the *installws* script:

---

ws.sqlsrv.server\_name

Indicates the name of the Adaptive Server Enterprise on which to run *installws*.

---

ws.sqlsrv.sa\_login

Indicates the user login for the Adaptive Server Enterprise.

---

ws.sqlsrv.sa\_password

Indicates the password for the Adaptive Server Enterprise.

---

## Specifying properties file entries

Because the backslash “\” symbol is used as an escape character, entries that use single backslash symbols are not interpreted correctly. For example:

```
com.sybase.ase.ws.interfaces = d:\sybase\ini\sql.ini
```

To work around this, escape the backslash with another backslash. For example:

```
com.sybase.ase.ws.interfaces = d:\\sybase\\ini\\sql.ini
```

You can also use forward slashes. For example:

```
com.sybase.ase.ws.interfaces = d:/sybase/ini/sql.ini
```





# SOAP and Adaptive Server Enterprise Datatype Mapping

This appendix documents SOAP and Adaptive Server Enterprise datatypes.

## Datatype mapping

The following sections describe datatype mappings between SOAP and Adaptive Server Enterprise with respect to the Web Services feature.

### SOAP-to-ASE datatype mappings

The following table shows SOAP datatypes and their corresponding types in Adaptive Server Enterprise. These are used when mapping RPC/encoded Web services to proxy tables in Adaptive Server Enterprise.

**Table C-1: SOAP and Adaptive Server Enterprise datatypes**

| SOAP datatype | Adaptive Server Enterprise datatype                               |
|---------------|-------------------------------------------------------------------|
| string        | varchar<br>Length depends on Adaptive Server Enterprise page size |
| boolean       | smallint                                                          |
| float         | real                                                              |
| double        | double precision                                                  |
| decimal       | float                                                             |
| duration      | datetime                                                          |
| dateTime      | datetime                                                          |
| time          | datetime                                                          |
| date          | datetime                                                          |
| gYearMonth    | datetime                                                          |

| <b>SOAP datatype</b> | <b>Adaptive Server Enterprise datatype</b>                        |
|----------------------|-------------------------------------------------------------------|
| gYear                | datetime                                                          |
| gMonthDay            | datetime                                                          |
| gDay                 | datetime                                                          |
| gMonth               | datetime                                                          |
| hexBinary            | Unsupported                                                       |
| base64Binary         | Unsupported                                                       |
| anyURI               | varchar<br>Length depends on Adaptive Server Enterprise page size |
| QName                | varchar<br>Length depends on Adaptive Server Enterprise page size |
| NOTATION             | varchar<br>Length depends on Adaptive Server Enterprise page size |
| normalizedString     | varchar<br>Length depends on Adaptive Server Enterprise page size |
| token                | varchar<br>Length depends on Adaptive Server Enterprise page size |
| language             | varchar<br>Length depends on Adaptive Server Enterprise page size |
| NMTOKEN              | varchar<br>Length depends on Adaptive Server Enterprise page size |
| Name                 | varchar<br>Length depends on Adaptive Server Enterprise page size |
| NCName               | varchar<br>Length depends on Adaptive Server Enterprise page size |
| ID                   | varchar<br>Length depends on Adaptive Server Enterprise page size |
| IDREF                | varchar<br>Length depends on Adaptive Server Enterprise page size |

| <b>SOAP datatype</b>   | <b>Adaptive Server Enterprise datatype</b>                        |
|------------------------|-------------------------------------------------------------------|
| ENTITY                 | varchar<br>Length depends on Adaptive Server Enterprise page size |
| integer                | integer                                                           |
| nonPositiveInteger     | integer                                                           |
| negativeInteger        | integer                                                           |
| long                   | integer                                                           |
| int                    | integer                                                           |
| short                  | smallint                                                          |
| byte                   | tinyint                                                           |
| nonNegativeInteger     | integer                                                           |
| unsignedLong           | integer                                                           |
| unsignedInt            | integer                                                           |
| unsignedShort          | smallint                                                          |
| unsignedByte           | tinyint                                                           |
| positiveInteger        | integer                                                           |
| soap arrays            | Not supported                                                     |
| soap complex datatypes | Not supported                                                     |

## ASE-to-SOAP datatype mappings for the *create service* command

The following table defines the relationship between datatypes available as a type to a stored procedure and the datatype used for a SOAP user-defined Web service.

**Note** For a Web service accessed through the HTTP GET method, which corresponds to Web services of type http or raw, all parameters are mapped to a type of xsd:string.

**Table C-2: SOAP and Adaptive Server Enterprise datatypes**

| ASE datatype       | SOAP parameter type |
|--------------------|---------------------|
| tinyint            | xsd:int             |
| smallint           | xsd:int             |
| int                | xsd:int             |
| bigint             | xsd:decimal         |
| numeric (p,s)      | xsd:decimal         |
| decimal (p,s)      | xsd:decimal         |
| float (precision)  | xsd:float           |
| double (precision) | xsd:double          |
| real               | xsd:double          |
| smallmoney         | xsd:int             |
| money              | xsd:int             |
| smalldatetime      | xsd:datetime        |
| datetime           | xsd:datetime        |
| date               | xsd:datetime        |
| time               | xsd:datetime        |
| char(n)            | xsd:string          |
| varchar(n)         | xsd:string          |
| unichar            | xsd:string          |
| univarchar         | xsd:string          |
| nchar              | xsd:string          |
| nvarchar           | xsd:string          |
| binary             | xsd:byte[]          |
| varbinary          | xsd:byte[]          |
| xml                | xsd:string          |
| text               | Not supported       |
| image              | Not supported       |
| Java Data Type     | Not supported       |

# Glossary

|                                |                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>ASE Web Services Engine</b> | The component of ASE Web Services that acts as a producer and consumer of Web services. The ASE Web Services Engine enables a client application to access Adaptive Server Enterprise stored procedures and SQL as Web methods. The ASE Web Services Engine also maps Web methods to proxy tables, allowing a client application to invoke the Web method through a SQL select statement. |
| <b>document/literal</b>        | A type of Web method for which communicating parties specify the data being transmitted and formatted according to XML schemas incorporated into the WSDL file.                                                                                                                                                                                                                           |
| <b>DTD</b>                     | Document Type Definition, used to define the legal building blocks of an XML document. A DTD can be declared within an XML document or referenced externally.                                                                                                                                                                                                                             |
| <b>HTTP</b>                    | Hypertext Transfer Protocol, part of the application layer of the Internet Protocol suite and is the primary means of exchanging information in the World Wide Web.                                                                                                                                                                                                                       |
| <b>HTTPS</b>                   | HTTP with SSL or TLS encryption.                                                                                                                                                                                                                                                                                                                                                          |
| <b>LDAP</b>                    | Lightweight Directory Access Protocol, an application-level protocol for Internet-based directory services.                                                                                                                                                                                                                                                                               |
| <b>RPC/encoded</b>             | A type of Web method invoked with SOAP messages containing an XML element for each method parameter.                                                                                                                                                                                                                                                                                      |
| <b>schema</b>                  | An outline defining the structure, content, and semantics of an XML document.                                                                                                                                                                                                                                                                                                             |
| <b>SDS</b>                     | Specialty Data Store, used as a Component Integration Service (CIS) to map a Web method to a proxy table.                                                                                                                                                                                                                                                                                 |
| <b>SOAP</b>                    | Simple Object Access Protocol, a standard for XML-based messaging across a network.                                                                                                                                                                                                                                                                                                       |
| <b>SQLX</b>                    | SQL/XML, an ANSI and ISO standard that provides support for using XML in the context of a SQL database system.                                                                                                                                                                                                                                                                            |

|                                 |                                                                                                                                                                                                                                                  |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>UDDI</b>                     | Universal Description Discovery and Integration.                                                                                                                                                                                                 |
| <b>URI</b>                      | Uniform Resource Identifier, a string of characters that identify an Internet Resource. The most common URI is the Uniform Resource Locator (URL), which identifies an Internet address. A less common URI is the Universal Resource Name (URN). |
| <b>URL</b>                      | Uniform Resource Locator, one type of URI.                                                                                                                                                                                                       |
| <b>User-defined Web service</b> | An arbitrary SQL statement named by the end user and executed with a SOAP client or through a Web browser.                                                                                                                                       |
| <b>W3C</b>                      | World Wide Web Consortium, which produces the software standards for the World Wide Web.                                                                                                                                                         |
| <b>Web method</b>               | A function described by WSDL and invoked through SOAP message.                                                                                                                                                                                   |
| <b>Web service</b>              | One or more Web methods described by a WSDL file.                                                                                                                                                                                                |
| <b>WSDL</b>                     | Web Services Description Language, which describes the public interface to a Web service.                                                                                                                                                        |
| <b>Xerces</b>                   | The Apache open-source XML parser.                                                                                                                                                                                                               |
| <b>XML</b>                      | Extensible Markup Language, a markup language standardized by W3C.                                                                                                                                                                               |
| <b>XML schema</b>               | A description of an XML document consisting of structure and content constraints.                                                                                                                                                                |
| <b>XPath</b>                    | XML Path Language, a language for addressing parts of an XML document.                                                                                                                                                                           |
| <b>XQL</b>                      | XML Query Language, a precursor of XQuery.                                                                                                                                                                                                       |

# Index

## A

- add option
  - parameters 40
  - syntax 40
  - usage 40
- addaliases option
  - parameters 51
  - syntax 51
  - usage 52
- Apache
  - sample client 59
- ASE
  - datatype mapping to SOAP 91
- ASE Web Services
  - advantages 4
  - configuration 23
  - directory tree 81
  - licensing 25
  - log files 56
  - logging 56
  - methods 36
  - security 26
  - using 33
- ASE Web Services Engine
  - as consumer 17
  - as producer 14
  - consumer components 17
  - HTTP handler 15
  - producer components 14
  - producer Web methods 15
  - proxy tables 19
  - SOAP handler 15
  - starting 33
  - user-defined Web services 16
  - using 33
  - XML mapper 15
- auditing
  - deploy option 55
  - undeploy option 55

user-defined Web services 54

## B

- bin directory 82

## C

- certificate for Microsoft .NET 28
- configssl 27
- configuration 23
  - files 25
  - logging.properties 25
  - logging.properties file 26
  - properties 85
  - ws.properties 25
  - ws.properties file 26
  - wsmmsg.properties file 26
- configuring
  - after installation 24
  - during installation 24
- create service command 46
  - examples 48, 49, 50
  - parameters 47
  - syntax 47

## D

- datatypes
  - ASE-to-SOAP mappings for create service 94
  - mapping 91
  - mapping between SOAP and ASE 91
  - SOAP and Adaptive Server Enterprise 91
- deploy option
  - auditing 55
  - parameters 52
  - syntax 52

## Index

- usage 52
- detailed logging 75
- diagnostic tools 74
  - detailed logging 75
  - JDBC-level tracing 75
- directories
  - bin 82
  - lib 82
  - logs 83
  - producer 83
  - props 84
  - samples 84
- document/literal Web methods 19
- drop service command 51
  - example 51
  - parameters 51
  - syntax 51
- dropalias option
  - parameters 53
  - syntax 53
  - usage 53

## E

- execute method 36
  - examples 38
  - parameters 36
  - syntax 36
- Extensible Markup Language 6

## F

- functions 4

## H

- handlers
  - HTTP 15
  - SOAP 15
- help option
  - parameters 41
  - syntax 41
  - usage 41

- HTTP handler 15
- http.log 56

## I

- installation
  - contents 81
- invoking a Web service 43
  - examples 43, 44

## J

- JDBC-level tracing 75

## L

- LDAP 5
- lib directory 82
- licensing 25
- list option
  - parameters 41
  - syntax 41
  - usage 42
- listalias option
  - syntax 53
  - usage 53
- listudws option
  - parameters 53
  - syntax 53
  - usage 53
- log files 56
- log4j 57
- logging 56
  - http.log 56
  - setting policies 57
  - webservices.log 56
- logging.properties file 26
- login method 38
  - parameters 38
  - syntax 38
  - usage 39
- logout method 39
  - syntax 39



usage 39  
logs directory 83

## M

messages 76  
methods 36  
    execute 36  
    login 38  
    logout 39  
Microsoft .NET  
    installing SSL certificate 28  
    sample client 64  
modify option  
    parameters 42  
    syntax 42  
myres.properties file  
    contents 87

## N

null passwords 72

## P

producer directory 83  
properties  
    com.sybase.ase.ws.consumer.cisport 85  
    com.sybase.ase.ws.consumer.logfilename 85  
    com.sybase.ase.ws.interfaces 86  
    com.sybase.ase.ws.libtcl 86  
    com.sybase.ase.ws.maxpostsize 86  
    com.sybase.ase.ws.producer.httpport 85  
    com.sybase.ase.ws.producer.jettylogfile 85  
    com.sybase.ase.ws.producer.ssl.httpport 87  
    com.sybase.ase.ws.producer.ssl.keypassword 87  
    com.sybase.ase.ws.producer.ssl.keystore 87  
    com.sybase.ase.ws.producer.ssl.password 87  
    com.sybase.ase.ws.producer.tuning.maxidletime  
        86  
    com.sybase.ase.ws.producer.tuning.maxthreads  
        86  
    com.sybase.ase.ws.producer.tuning.minthreads

86  
com.sybase.ase.ws.producer.tuning.ssl.maxidletime  
    86  
com.sybase.ase.ws.producer.tuning.ssl.maxthreads  
    86  
com.sybase.ase.ws.producer.tuning.ssl.minthreads  
    86  
com.sybase.ase.ws.ui.activate 86  
ws.consumer.host 88  
ws.consumer.log 88  
ws.consumer.name 88  
ws.consumer.port 88  
ws.ini 87  
ws.libtcl 87  
ws.producer.jettylogfile 87  
ws.producer.log 87  
ws.producer.port 87  
ws.sqlsrv.sa\_login 88  
ws.sqlsrv.sa\_password 88  
ws.sqlsrv.server\_name 88  
ws.ssl.certificatepassword 88  
ws.ssl.host 88  
ws.ssl.keystorelocation 88  
ws.ssl.keystorepassword 88  
properties files  
    specifying entries 89  
props directory 84  
proxy tables 19

## R

remove option  
    parameters 42  
    syntax 42  
RPC/encoded Web methods 19  
runws failure 72

## S

sample applications 59  
    Apache client 59  
    Execute.exe 64  
    Microsoft .NET client 64  
    runexecute 60

## Index

- samples directory 84
- security 5, 26
  - user-defined Web services 54
- Simple Object Access Protocol, *See* SOAP 11
- SOAP 11
  - datatype mapping to ASE 91
  - message structure 11
  - Web services standard 11
- SOAP handler 15
- SOAP stack 18
- sp\_webservices
  - add option 40
  - addalias option 51
  - deploy option 52
  - dropalias option 52
  - help option 41
  - list option 41
  - listalias option 53
  - listudws option 53
  - modify option 42
  - remove option 42
  - undeploy option 53
  - using 39
  - with user-defined Web services 51
- SQL 5
- SSL
  - configuring 27
- stopws failure 72
- stored procedures 4
- Sybase Central 57

## T

- troubleshooting issues 69
  - abnormal termination of sp\_webservices 73
  - command-line arguments 72
  - entries in ws.properties 71
  - locating WSDL 71
  - null passwords 72
  - remote server class definition setting 69
  - runws or stopws fails 72
  - specifying SOAP endpoints with SSL 73
  - starting the ASE Web Services Engine 70
  - sysattributes table entry 74
  - truncated document/literal results 70

- unmapped RPC/encoded Web method 70
- Web Services proxy table restrictions 73

## U

- undeploy option
  - auditing 55
  - parameters 53
  - syntax 53
  - usage 54
- user-defined Web services 5
  - auditing 54
  - commands 46
  - security 54
  - using 46
  - with sp\_webservices 51
- using ASE Web Services 33

## W

- Web methods
  - document/literal 19
  - RPC/encoded 19
- Web services
  - invoking 43
  - overview 3
  - standards 6
- Web Services Description Language, *See* WSDL 9
- webservice.log 56
- ws.properties file 26
  - contents 85
  - entries 71
- WSDL 9
  - locating 71
  - Web services standard 9
- wsmsg.properties file 26

## X

- XML 6
  - document structure 7
  - Web services standard 6
- XML mapper 15