ADMINISTRATION GUIDE | PUBLIC
SAP Adaptive Server Enterprise 16.0 SP03
Document Version: 1.0 – 2019-06-06

Web Services Users Guide

THE BEST RUN **SAP**

Content

1 Understanding SAP Adaptive Server Enterprise Web Services

Web Services consists of the SAP ASE Web Services Engine that runs independently of SAP ASE.

The Web Services Engine provides the following functionality:

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

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).

1.1 Advantages of SAP ASE Web Services

With the SAP 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 SAP ASE Web Services Engine to manage data in SAP ASE. SQL can also be used to invoke Web services through the SAP ASE Web Services Engine.

Security

The SAP 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. SAP ASE Web Services enables client applications to access Web methods using LDAP.

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

User-Defined Web Services

Using user-defined Web services, you can execute SQL commands in SAP ASE 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 SAP ASE that is compliant with SOA.

1.2　Web Services Standards

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

1.2.1　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.

1.2.1.1　XML Example

An example of an `isql` query is provided.

The query is on the `pubs2` database to find 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 SAP 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.

1.2.1.2   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 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/

1.2.2 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 SAP 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 SAP 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 SAP ASE Web Services Engine is running,
- `<producer_port>` – is the port number.

1.2.3 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.

1.2.3.1 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.

See the following for detailed information on SOAP:

- Simple Object Access Protocol (1:1) ↗
- Simple Object Access Protocol (1.2) Part 1 ↗
- Simple Object Access Protocol (1.2) Part 2 ↗

2 Understanding the SAP ASE Web Services Engine

The SAP ASE Web Services Engine provides the producer of Web services and consumer of Web services functionality.

Related Information

2.1 Producer of Web Services

The SAP ASE Web Services Engine acts as a producer of Web services by enabling a client application to access SAP ASE stored procedures and SQL using SOAP.

The output of the SAP ASE Web Services Engine complies with SQLX, which is defined as part of the ANSI specification for SQL.

> i Note
>
> SAP recommends that you run the SAP ASE Web Services Engine on the same machine as SAP ASE.

This diagram shows the SAP ASE Web services engine for client access to SAP ASE.
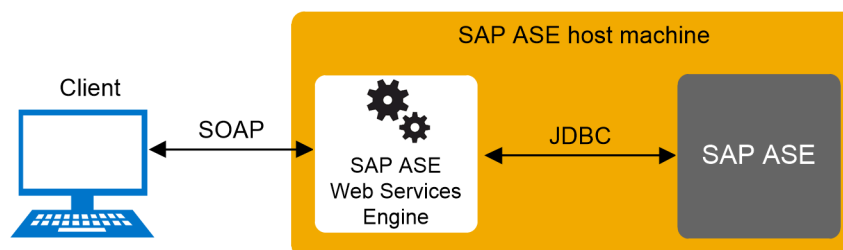
The client can send a SQL or stored procedure command as a SOAP request, and any result is returned as a SOAP response. The data in the SOAP response conforms to the SQLX standard.

2.1.1 Producer Components

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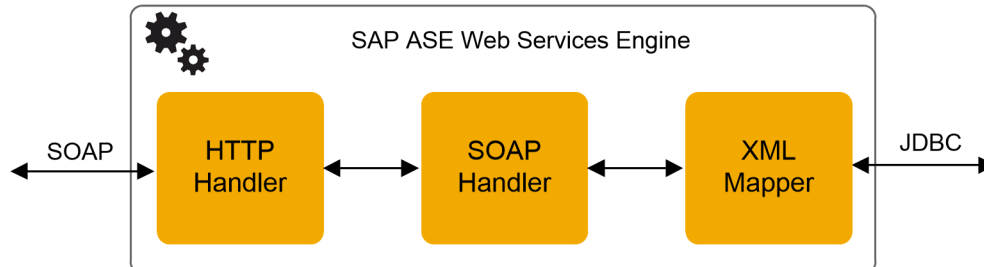
HTTP Handler

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.

> **i 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 SAP ASE 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.

2.1.2 Producer Web Methods

The SAP ASE Web services engine provides the `execute`, `login`, and `logout` methods.

- `execute` – executes a SQL statement or stored procedure.

- `login` – establishes a persistent connection to SAP ASE.
- `logout` – explicitly terminates an SAP ASE connection.

Related Information

Using SAP ASE Web Services [page 25]

2.1.3 User-Defined Web Services

In addition to the Web methods provided by the SAP ASE Web Services Engine, SAP ASE Web Services enables you to create Web services and execute SQL commands in SAP ASE using either a Web browser or a SOAP client.
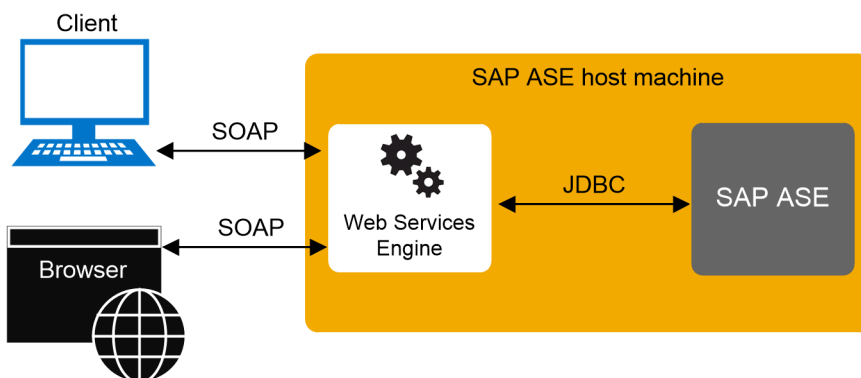
You can create a user-defined Web service with the `create service` command, which enables you to specify the SQL to be executed, create a first-class object for which permissions can be controlled with the `grant` command, and control whether the service can be invoked with a Web browser or a SOAP client. The SAP ASE Web Services Engine automatically generates WSDL for user-defined Web services. For details on creating and using user-defined Web services, see *Using SAP ASE Web Services*.

> i 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.

Related Information

Using SAP ASE Web Services [page 25]

2.2 Consumer of Web Services

The SAP ASE Web Services Engine acts as a consumer of Web services by enabling SAP ASE to access and execute Web methods.

A Web method is made accessible by mapping it to an SAP ASE 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.
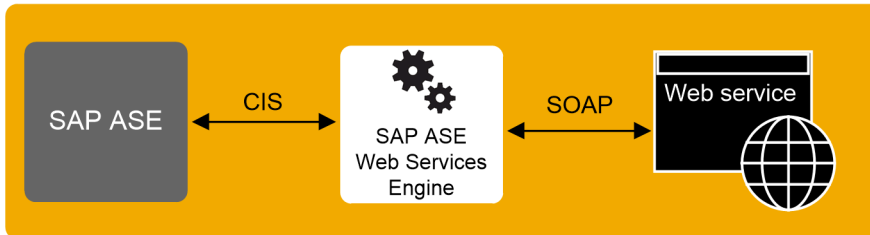
> **i Note**
>
> The Web service accessed may reside within or beyond a firewall.

2.2.1 Consumer Components

Acting as a consumer of Web services, the SAP 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.
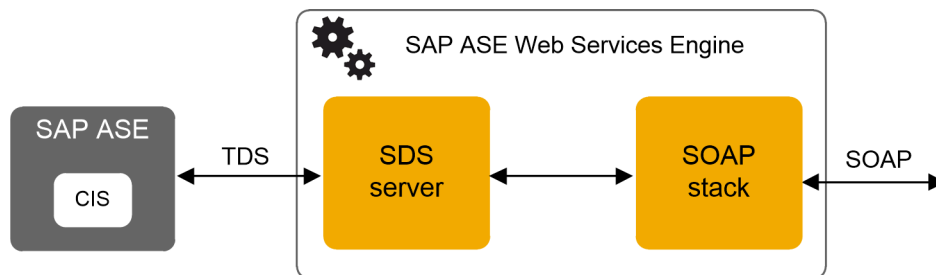
2.2.1.1 SDS Server

When SAP ASE receives a `select` statement for a Web method proxy table, SAP ASE forwards the request to the SAP ASE Web Services Engine in a Tabular Data Stream™ (TDS).

The SDS server, which acts as a server for CIS, allows the SAP ASE Web Services Engine to intercept and handle the TDS from SAP ASE.

2.2.1.2 SOAP Stack

The SOAP stack is a layered set of functionality 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 SAP ASE 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.

> **i 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 SAP ASE.
>
> 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. SAP supports this recommendation.

2.2.2 Proxy Tables

Because they point to Web methods, Web Services proxy tables are of type `procedure` and are subject to restrictions.

The restriction types are:

- 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*.

2.3 Web Services Engine as a Consumer Versus Producer

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

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

3 Installing and Configuring SAP ASE Web Services

SAP ASE Web Services is installed as part of the installation for SAP ASE.

SAP ASE Web Services may be configured during the installation of SAP ASE using a configuration wizard or after the installation. For instructions on installing SAP ASE, see the *Installation Guide*.

> **i Note**
>
> Unless otherwise specified, directories listed in this and subsequent chapters are assumed to reside under the `$SYBASE/WS-16_0` directory on UNIX and the `%SYBASE%\WS-16_0` directory on Windows.

3.1 Installing and Configuring Web Services

Install and configure Web Services.

Context

You have the option of configuring SAP ASE Web Services during a custom installation. To configure Web Services, activate the Configuration Utility during the installation of the Web Services feature, and follow the steps in the wizard.

Procedure

1. Add an entry to the interfaces file for the Web Services Consumer. The interfaces file is located in `$SYBASE` (or `%SYBASE%` on Windows).
2. If necessary, update the `ws.properties` file. The `ws.properties` file contains a number of runtime properties for the Web Services server, including the server name, port number, error logging, and so on. The `ws.properties` file contains directions for updating in its commented section. Update the values in the file to suit your installation.

   The default location for `ws.properties` is:
   - On UNIX – `ws.properties` is located in `$SYBASE/WS-16_0/props/`
   - On Windows – `ws.properties` is located in `%SYBASE%\WS-16_0\props\`

3. Run the `configssl` utility to configure SSL for Web Services: The syntax is:

```
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 used to connect to Web Services using SSL. For example, the `<domain_hostName>` for the `mydomainhostname` URL is:

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

  `<domain_hostName>` has no default value.
- `<keystore>` – full path to the file that stores certificates. The default locations are:
    - On UNIX – `$SYBASE/WS-16_0/props/keystore`
    - On Windows – `%SYBASE%\WS-16_0\props\keystore`
- `<httpsPort>` – the port number to listen for an SSL connection. The default port number is 8182.
- `<property_file>` – the location and name of the `properties` file to update. The default locations are:
    - On UNIX – `$SYBASE/WS-16_0/props/ws.properties`
    - On Windows – `%SYBASE%\WS-16_0\props\ws.properties`
- `<certificate_password>` – the password for the certificate. `<certificate_password>` has no default value. `configssl` prompts for a password if you do not provide one.
- `<keystore_password>` – the password for the `<keystore>`. `<keystore_password>` has no default value. `configssl` prompts for a password if you do not provide one.

This example configures SSL for Web Services on host name "sybase":

```
/sybase/ase160/WS-16_0/bin/configssl -d asekernel1 -c sybase -s sybase  -k /
sybase/ase160/WS-16_0/producer/keystore  -h  8187  -f /sybase/ase160/WS-16_0/
props/ws.properties
```

```
configssl initiating execution at Thu Jan 19 19:43:59 PST 2014.
Using SYBASE as /sybase/ase160
Using SYBASE_WS as WS-16_0
Using SYBASE_JRE6 as /sybase/ase160/shared/JRE-6_0_24_64BIT
Generating 1,024 bit RSA key pair and self-signed certificate (SHA1withRSA)
with a validity of 360 days
        for: CN=asekernel1, OU=ASEWS, O=Sybase, L=Boulder, ST=CO, C=US
[Storing /sybase/ase160/WS-16_0/producer/keystore]
Certificate stored in file </sybase/ase160/WS-16_0/producer/wscertificate.cer>
\nUpdating /sybase/ase160/WS-16_0/props/ws.properties file to reflect new SSL
settings.
Succeed to set permission mode as '600' for file '/sybase/ase160/WS-16_0/
props/ws.properties'
Update of /sybase/ase160/WS-16_0/props/ws.properties complete.
configssl execution complete at Thu Jan 19 19:44:02 PST 2014
```

4. If necessary, run `installws`. If Adaptive Server includes the `sp_webservices` stored procedure, `installws` was run previously, and you need not run it again.
5. If necessary, add a server entry to `sysservers` for Web Services:

```
sp_addserver '<web_services_name>, 'sds', '<web_services_name>'
```

For example, to add a server entry for a Web Services named 'ws:'

```
sp_addserver 'ws',sds,'ws'
```

3.2 Licensing

License entry for SAP ASE Web Services is handled by the InstallShield installation for SAP ASE.

Context

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

Procedure

1. Establish an `isql` session with your SAP ASE server.
2. Activate SAP ASE Web Services by entering the following command in `isql`:

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

3.3 Configuration Files

The `props` directory contains the `ws.properties`, `logging.properties`, and `wsmsg.properties file` configuration files.

- `ws.properties` – contains configuration settings for SAP ASE Web Services. For details on these configuration properties, see *Configuration Properties*.
- `logging.properties` – The 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*.
  The `logging.properties` file entries and logging behavior follow the Apache log4j package. For detailed information, refer to the documentation for log4j at http://jakarta.apache.org/log4j/docs/documentation.html.
- `wsmsg.properties file` – is for internal use by SAP ASE Web Services.

Related Information

3.4 Security

To ensure secure operation of SAP ASE Web Services, SAP recommends that you install SAP ASE Web Services on the same machine as SAP ASE and use SSL to connect to the SAP ASE Web services engine.

SAP ASE Web Services supports all authorization measures supported by SAP ASE.

Related Information

3.4.1 Configuring SSL

SSL is configured automatically using the Configuration Utility from InstallShield.

However, you can also configure SSL manually. To manually configure SSL for SAP ASE Web Services, run the `configssl` script, which can be found in the `bin` directory:

> **i Note**
>
> Two certificate passwords are created by default as "sybase" from InstallShield during installation.

```
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 `$SYBASE/WS-16_0/props/keystore`, or `%SYBASE%\WS-16_0\props\keystore` for Windows.
- `<httpsPort>` – is the port on which to listen for an SSL connection. The default is 8182.
- `<property_file>` – is the location and name of the properties file to update. The default location for UNIX is `$SYBASE/WS-16_0/props/ws.properties`, or `%SYBASE%\WS-16_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.

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 `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_JRE7_64/lib/security/cacerts`

You can use the following properties to disable specific cipher suites and TLS protocol to manage security of SSL.

- `com.sybase.ase.ws.producer.ssl.exciphersuites` – specifies the weak cipher suites to disable to prevent SSL attacks and browser errors.
- `com.sybase.ase.ws.producer.ssl.exprotocols` – specifies the TLS protocol versions to disable to prevent SSL attacks and browser errors.

For more information, see

3.4.2 Installing a Certificate for Microsoft .NET

A Microsoft .NET client requires a certificate to access the SAP ASE Web Services Engine using SSL.

Context

Use the following procedure to install a certificate for Microsoft .NET.

Procedure

1. Start the SAP ASE Web Services Engine with SSL. See *Starting and Stopping the SAP ASE Web Services Engine*.
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 SAP ASE Web Services Engine runs.
   - `<SSL_port>` – is the port for the SAP 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 SAP ASE Web Services."

3.4.3 Verifying the Certificate Installation

Verify the installation of a certificate for Microsoft .NET.

Procedure

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 SAP ASE Web Services Engine runs.
- `<SSL_port>` – is the port for the SAP ASE Web Services Engine.

No Security Alert dialog box should appear.

3.5 Configuring Web Services to Use a Proxy Server

SAP ASE web services can be configured to access external web services by defining the host name and port number of the proxy server in the script that is used to start web servers.

UNIX

On UNIX, the default `runws` script contains a line that defines certain parameters for web services such as the installation directory of the SAP ASE server and the Open Client installation directory:

```
ADDDEFINES="-Dsybase.home=$SYBASE -Docs.home=$SYBASE_OCS -
Dcom.sybase.ase.ws.libtcl=$SYBASE_TCL_CFG"
```

To enable access to external web services via an http proxy on host `gateway` listening on port `8888` amended this line with the proxy details as shown below:

```
ADDDEFINES="-Dsybase.home=$SYBASE -Docs.home=$SYBASE_OCS -
Dcom.sybase.ase.ws.libtcl=$SYBASE_TCL_CFG" -Dhttp.proxySet=true -
Dhttp.proxyHost=gateway1 -Dhttp.proxyPort=8888
```

Windows

On Windows, the default `runws.bat` script contains a line that defines certain parameters for web services such as the installation directory of the SAP ASE server and the Open Client installation directory:

```
set ADDDEFINES=-Dsybase.home="%SYBASE%" -Docs.home="%SYBASE_OCS%" -
Dcom.sybase.ase.ws.libtcl="%SYBASE_TCL_CFG%"
```

To enable access to external webservices via an http proxy on host `gateway` listening on port `8888`, amended this line with the proxy details as shown below:

```
set ADDDEFINES=-Dsybase.home="%SYBASE%" -Docs.home="%SYBASE_OCS%" -
Dcom.sybase.ase.ws.libtcl="%SYBASE_TCL_CFG%" -Dhttp.proxySet=true -
Dhttp.proxyHost=gateway1 -Dhttp.proxyPort=8888
```

4 Using SAP ASE Web Services

Using SAP ASE Web services includes: Starting and stopping the Web Services engine, invoking Web Services, understanding the Web Services methods, and using `sp_webservices`.

Before using ASE Web Services, make sure you have completed the configuration tasks.

Related Information

4.1 Starting and Stopping the SAP 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 SAP ASE server. 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 SAP ASE server. 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-16_0/props/ws.properties`, or `%SYBASE%\WS-16_0\props\ws.properties` for Windows.
- `-v` specifies that the ASE Web Services Engine should display version information at startup or shutdown.

4.1.1 Conditions

The ASE Web Services Engine will start or stop if certain 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 SAP ASE.
  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 SAP ASE.

  > ### i 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.

  > ### i Note
  >
  > The password for an SAP ASE server user cannot be set to a null string. The `sa` login allows null-string passwords by default. SAP does not recommend using null passwords.

- The login to SAP ASE has `sa` role privileges.
- The following stored procedure command has been executed in `isql` for your SAP ASE server:

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

4.1.2 Verifying that SAP ASE Web Services is Enabled and Running

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

Procedure

1. To verify that SAP ASE Web Services is enabled, execute the following command on SAP ASE:

   ```
   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.

2. Check the `producer.log` or `consumer.log` file in the `logs` directory for messages indicating that the SAP 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: ...
```

Results

> ### i Note
>
> The `runproducer`, `stopproducer`, `runconsumer`, and `stopconsumer` scripts are available in releases 15.0 and earlier. However, in release 15.0 and later, these scripts call the `runws` and `stopws` scripts.

4.2 SAP ASE Web Services Methods

To access SAP ASE Web Services, your client must use the methods exposed by the SAP 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 SAP ASE Web Services Engine provides the following methods:

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

These methods are supported by SAP ASE 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.

Related Information

execute [page 28]
login [page 30]
logout [page 30]

4.2.1 execute

The `execute` method executes a Transact-SQL statement or stored procedure in SAP ASE.

Syntax

```
execute <aseServerName userName password > <sqlxOptions> <sql>
```

Parameters

- `<aseServerName>`
  SOAP `string` indicating the name of the SAP ASE 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 SAP ASE Web Services* for details.
- `<userName>`
  SOAP `string` indicating the user ID needed to log in to the SAP ASE server.
- `<password>`
  SOAP `string` indicating the password needed to log in to the SAP ASE server.
- `<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*.

- `<sql>`

  SOAP `string` indicating the SQL statement or stored procedure to be executed on SAP ASE. 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 *Configuration Properties*.

Example 1

Checks the version number for SAP ASE.

```
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.

> i Note
>
> You must provide the `<userName>` and `password` parameters to invoke the `execute` method. SAP ASE verifies it can execute the specified SQL statement.

Example 2

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"
```

Related Information

Starting and Stopping the SAP ASE Web Services Engine [page 25]
Configuration Properties [page 73]

4.2.2 login

The `login` method establishes a persistent connection to SAP ASE.

Syntax

```
login <aseServerName userName password>
```

Parameters

- `<aseServerName>`
  SOAP `string` indicating the name of the SAP ASE server on which to execute the SQL statement or stored procedure.
  For the `login` method, SAP 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 SAP ASE server.
- `<password>`
  SOAP `string` indicating the password needed to log in to the SAP ASE server.

Usage

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

4.2.3 logout

The `logout` method terminates a persistent connection to SAP ASE.

Syntax

```
logout
```

Usage

The `logout` method terminates a persistent connection to SAP ASE established by the `login` method.

4.3 Using sp_webservices

The `sp_webservices` stored procedure creates and manages the proxy tables used in the SAP 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*.

> i Note
>
> For information on restrictions for Web Services proxy tables, see *Web Services Proxy Table Restrictions* in *Troubleshooting Issues*.

Related Information

Using sp_webservices with User-Defined Web Services [page 43]
Troubleshooting Issues [page 59]
add [page 32]
help [page 33]
list [page 34]
modify [page 34]
remove [page 35]
Web Services Proxy Table Restrictions [page 64]

4.3.1 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 SAP 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.

> **i 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`.

Related Information

[SOAP and SAP ASE Datatype Mapping [page 78]](#)

4.3.2 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.

4.3.3 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 SAP 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.

4.3.4 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.

4.3.5 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 SAP 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.

4.4 Invoking a Web Service

Invoke a Web Service by starting the Web Services Engine and invoking the Web method with a `select` statement.

Procedure

1. Start the SAP ASE Web Services Engine.
2. Use the `add` option of `sp_webservices` to map the Web service to a proxy table in SAP ASE.
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.

4.4.1 Examples of Invoking a Web Service

Examples of invoke a Web service using the SAP ASE Web Services Engine.

Example 1

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><Symb
ol>SY</Symbol><Last>21.48</Last><Date>7/21/2005</Date><Time>4:01pm</Time><Cha
nge>+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><Earns>0.706</Earns><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 ,stocksymbol
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)
```

4.5   Using User-defined Web Services

The functionality specific to user-defined Web services includes: Commands, using sp_webservices, security, and auditing.

4.5.1 Commands for User-Defined Web Services

You can create, drop, and alter user-defined Web services using the commands `create service` and `drop service`.

4.5.1.1   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.

> **i Note**
>
> To make a user-defined Web service available through the SAP 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* in *Using sp_webservices*.

Syntax

```
create service <service-name> [secure <security_options> ] [, userpath <path>]
          [, alias <alias-name>]
      type { xml | raw | soap }
      [[(@<parameter_name> datatype [(length ) | (precision  [, scale ])]
          [= default][output]
```

```
        [, @<parameter_name> datatype [(length ) | (precision  [, scale ])]
            [= default][output]]...[)]]
as< SQL_statements>
```

```
<security_options> ::= (<security_option_item> [<security_option_item>])
```

Parameters

- <service-name> – 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.
- <security_option_item> – either clear or ssl:
  - clear indicates that HTTP is used to access this Web service.
  - ssl indicates HTTPS is used to access this Web service
- <path> – 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.
- <alias-name> – a character-string-literal specifying the user-defined Web service alias.
- <parameter_name> – 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 <parameter_name> parameter of the create procedure command.
- <SQL_statements> – 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 <SQL_statements> parameter of the create procedure command.
- type – can be soap, raw, or xml:
  - 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.

  > i Note
  >
  > For datatype mappings between SAP ASE stored procedures and SOAP user-defined Web services, see SAP ASE-to-SOAP Datatype Mappings for the create service Command in SOAP and SAP ASE 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-16_0/producer` (`%SYBASE%\WS-16_0\producer` on Windows), you can access the page https://myhost:8182/testraw.html, where the `<username>`, `<password>`, and `<database>` are `bob`, `bob123`, and `pubs2`, respectively. Click "Access rawservice" to display the result.

> **i Note**
>
> For details on the `deploy` option for `sp_webservices`, see *Using sp_webservices with User-Defined Web Services*.

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 SAP ASE version string, is displayed in an HTML `<h1>` tag in the browser window:

```html
<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"></p>
     </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
```

> i Note
>
> For details on the `deploy` option for `sp_webservices`, see *Using sp_webservices with User-Defined Web Services*.

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 the HTML file to `$SYBASE/WS-16_0/producer` (`%SYBASE%\WS-16_0\producer` on Windows). Access the page 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"></p>
   </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)`.

Related Information

Using sp_webservices [page 31]
SOAP and SAP ASE Datatype Mapping [page 78]
Using sp_webservices with User-Defined Web Services [page 43]
deploy [page 45]
SAP ASE-to-SOAP Datatype Mappings for the create service Command [page 80]

4.5.1.2   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.

> i 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*.

Syntax

```
drop service <service-name>
```

Parameters

- `<service-name>` – 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 an service that does not exist, an exception results. Also, you cannot drop a service that is currently in use by another session.

Example

This example drops the user-defined Web service named `sp_who_service`:

```
drop service sp_who_service
```

Related Information

Using sp_webservices with User-Defined Web Services [page 43]

4.5.2 Using sp_webservices with User-Defined Web Services

Several `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.

Related Information

4.5.2.1 addalias

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

Syntax

```
sp_webservices 'addalias' <alias_name> , <database_name>
```

Parameters

- `<alias_name>`
  The alias for the specified database. This parameter is required. If you define an alias name for a specific database, the producer displays the alias name instead of the database name. If you have not defined an alias name, Webservices uses `dboralias` as the database name.
- `<database_name>`
  The name of the database for which the alias is specified. This parameter is required.

Examples

Adds `marketing_alias` as the alias name for database `marketing_db`:

```
sp_webservices 'addalias', marketing_alias, marketing_db
```

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.

4.5.2.2　deploy

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

> **i 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 SAP ASE Web Services Engine deletes its internal cache of user-defined Web services and rereads all metadata about user-defined Web services from SAP ASE.

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

4.5.2.3   dropalias

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

Syntax

```
sp_webservices 'dropalias' <alias_name>
```

Parameters

- `<alias_name>`
  The alias to be dropped.

Example

Drops the `marketing_alias` alias:

```
sp_webservices 'dropalias', marketing_alias
```

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.

4.5.2.4   listudws

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

Syntax

```
sp_webservices 'listudws' [, '<service_name>']
```

Parameters

- `<service_name>`
  The name of the user-defined Web service to be listed.

Usage

If the `<service_name>` parameter is not specified, all user-defined Web services are listed.

4.5.2.5   listalias

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

Syntax

```
sp_webservices 'listalias'
```

Usage

All aliases are listed.

4.5.2.6   undeploy

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

Syntax

```
sp_webservices 'undeploy', ['all' | '<service_name>']
```

Parameters

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

4.5.3 Security for User-Defined Web Services

The system role `webservices_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 SAP ASE. See the *Security Administration Guide* for details on how to set the proper privileges using the `grant` and `revoke` commands.

> i Note
>
> For the most current information on security in SAP ASE Web Services, see the *Release Bulletin*.

4.5.4 Auditing for User-Defined Web Services

User-defined Web services are modeled as stored procedures within SAP ASE.

In manipulating user-defined Web services, SAP ASE 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 *Security Administration Guide*.

In addition to existing auditing functionality, SAP ASE 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"
```

4.5.4.1 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 *Security Administration Guide*.

4.5.4.2 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 *Security Administration Guide*.

4.6 SAP ASE Web Services Logging

By default, SAP ASE Web Services logs only informational and error messages.

SAP 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

For details on how to log more detailed information, contact SAP Product Support.

4.6.1 Rolling Over Log Files

Logging is implemented in SAP ASE Web Services using the Apache log4j framework.

For information on specific log4j parameters and implementing a rollover policy, see the Apache Web site .

5   Sample Applications

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

5.1   Apache Sample Client

The Apache sample client and script are found in the `$SYBASE/WS-16_0/samples/apacheclient` directory in UNIX, or in the `%SYBASE%\WS-16_0\samples\apacheclient` directory in Windows.

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

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

5.1.1   Using runexecute

The `runexecute` script executes a stored procedure or Transact-SQL statement on SAP ASE through SAP 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 SAP ASE server in the `interfaces` file or LDAP server.
- `<user_ID>`
  The user ID needed to log in to the SAP ASE server.
- `<password>`
  The password needed to log in to the SAP ASE server.
- `<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*.
- `<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 SAP ASE. This statement must be delimited by double quotes.

Example 1

This example checks the version number for SAP ASE using a `select` statement.

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

SAP 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"
```

SAP 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/svr/xmltestdir/resultset.xsd',targetns='htt
p://www-edm/remote/svr/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)"
```

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

5.2 Microsoft .NET Sample Client

The Microsoft .NET sample client and script are found in the `$SYBASE/WS-16_0/samples/ms.net/Execute/bin/Release` directory in UNIX, or in the `%SYBASE%\WS-16_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

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

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. This script can be found in the `Release` directory.

5.2.1 Using Execute.exe

`Execute.exe` executes a stored procedure or Transact-SQL statement on SAP ASE through SAP 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 SAP ASE server in the `sql.ini` file or LDAP server.
- `<user_ID>`
  The user ID needed to log in to the SAP ASE server.
- `<password>`
  The password needed to log in to the SAP ASE server.
- `<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*.

- `<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 SAP ASE. This statement must be delimited by double quotes.

Example 1

This example checks the version number for SAP ASE.

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

SAP 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"
```

SAP 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/svr/xmltestdir/resultset.xsd',targetns='htt
p://www-edm/remote/svr/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)"
```

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

6 Troubleshooting Issues

Descriptions of common issues, messages, and diagnostic tools are provided to assist you in troubleshooting SAP ASE Web Services.

6.1 Remote Server Class Definition Setting

The `sp_webservices add` command may return an error when generating proxy tables.

This error occurs because the remote server representing the SAP ASE Web Services Engine has been added using `sp_addserver` with a class other than "sds."

```
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)
```

To verify that this is so, use `sp_helpserver` in `isql`:

```
sp_helpserver <ws>
```

Here, `<ws>` is the name of the SAP 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 SAP ASE Web Services Engine.

6.2   Unmapped RPC/Encoded Web Method

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 SAP ASE.

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.

6.3   Truncated Document/Literal Results

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 SAP ASE when text or image data is truncated with a `select` command. However, a warning is logged for the SAP ASE Web Services Engine, so check your `consumer.log` file.

6.4   Starting SAP ASE Web Services Engine

The `runws` script does not successfully start a SAP 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. SAP ASE Web Services requires JRE 1.8 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 SAP ASE Web Services Engine with a different properties file:

   ```
   C:\sybase\WS-16_0\bin\runws -f
   C:\sybase\WS-16_0\props\myfile.properties
   ```

Issue

The SAP 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
```

6.5   Locating WSDL

A client connecting to the SAP ASE Web Services Engine through a Web browser cannot find the WSDL file, or the SAP ASE Web Services Engine cannot find the WSDL file to perform proxy-table mapping.

User Action

Verify that the SAP ASE Web Services Engine is running. If you are using the SAP ASE Web Services Engine through a browser, make sure the browser URL indicates `https://` for an SSL connection and `http://` for a standard connection.

6.6 Specifying Entries in ws.properties

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
```

6.7 Windows NT Command-Line Arguments

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
```

6.8 Run or Stop Scripts Fail

The `runws` or `stopws` script fails to execute.

User Action

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

- Verify that your SAP ASE is running.
- Make sure that the user name and password you specified are valid to log in to your SAP ASE.
- 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 SAP ASE.

  > i Note
  >
  > On Windows systems, the `interfaces` file is named `sql.ini`.

- Verify that the user name has `<sa_role>` privileges.

6.9 Null Passwords

The password for an SAP ASE 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.

6.10　Specifying SOAP Endpoints with SSL

SAP ASE Web Services methods or sample applications do not return results with the `<aseServerName>` specified at invocation.

User Action

Make sure the `<aseServerName>` 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 `<aseServerName>` is the same name you supplied in creating an SSL certificate with the `configssl` script. For example:

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

Here, the value of `<aseServerName>` supplied when invoking an SAP 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.

6.11　Abnormal Termination of sp_webservices 'add'

Proxy tables created during execution of the `sp_webervices` 'add' option remain after an abnormal termination of `sp_webservices`, as with a Ctrl+C interrupt or an SAP ASE crash.

User Action

Use the `remove` option to delete any proxy tables created by the `add` option when `sp_webservices` terminated abnormally.

6.12　Web Services Proxy Table Restrictions

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)
```

6.13   sysattributes Table Entry

An 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 SAP ASE server 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 SAP ASE server from your `interfaces` or `sql.ini` file. Then, restart SAP ASE.

6.14 Messages

Messages for SAP ASE Web Services.

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 SAP ASE and the SAP ASE Web Services Engine.\nDetails: [%s].
15205	Caught SQLException. This usually indicates an error retrieving meta data from SAP 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 SAP 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 (MalformedURL) 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].

Message number	Message Text
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.
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 SAP 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.

Message number	Message Text
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.

6.15 Diagnostic Tools

SAP 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 SAP ASE Web Services Engine.

The following is an example of the content of the `logging.properties` file as it is delivered with SAP 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.

7 SAP ASE Web Services Directory Tree

The contents of the SAP ASE Web Services installation is installed at the same level as the root directory for SAP ASE. The SAP ASE Web Services root directory is named `WS-16_0` and consists of several subdirectories.

Table 1: SAP ASE Web Services Directories

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

Table 2: bin Directory Contents

File/Directory name	Function
configssl	Configures SSL.
getpass.exe	Used for SAP ASE login. This file is only available on Windows.
installws	Installs `sp_webservices` stored procedure.
runtcpmon	Creates a monitor to trace SOAP messages.
runws	Starts the SAP ASE Web Services Engine.
stopws	Stops the SAP ASE Web Services Engine.

> i Note
>
> The Windows versions of these files have a `.bat` suffix.

Table 3: lib Directory Contents

File/Directory name	Function
axis.jar	Apache Axis file

File/Directory name	Function
commons-discovery.jar	Apache Axis file
commons-logging.jar	Apache Axis file
javax.servlet.jar	Servlet library
jaxrpc.jar	Apache Axis file
log4j-1.2.4.jar	log4j logger
mail.jar	Apache Axis file
org.mortbay.jetty.jar	HTTP server
saaj.jar	Apache Axis file
sqlx.jar	SQLX file
tools.jar	Java tools (for example, the javac compiler)
ws.jar	Code for SAP ASE Web Services Engine
ws_debug.jar	Debugger Code for SAP ASE Web Services Engine
wsdl4j.jar	Apache Axis file
xercesImpl.jar	Xerces parser
xmlParserAPIs.jar	Xerces parser

Table 4: logs Directory Contents

File/Directory name	Function
webservices.log	contains information and error messages for Web Services producers and consumers
http.log	Logs Web server activity.

Table 5: producer Directory Contents

File/Directory name	Function
index.html	Home page for the Web services browser interface.
keystore	Holds encryption keys.
ui	Directory structure for Web services browser interface.

File/Directory name	Function
WEB-INF	Directory structure required for Jetty.
wscertificate.cer	Auto-generated certificate for SSL. This file is present only the `configssl` script has been run.

Table 6: props Directory Contents

Directory name	Function
logging.properties	Configuration file for log4j
ws.properties	All configuration parameters for SAP ASE Web Services Engine
wsmsg.properties	Configuration file for messages

Table 7: samples Directory Contents

Directory name	Function
apacheclient	Directory containing sample scripts for compiling and running the sample client
ms.net	Directory containing samples for .NET

> i Note
>
> The samples directory contains precompiled and source code for both Apache and Microsoft .NET. You can use source code in your own applications.

8 Configuration Properties

SAP ASE Web Services configuration properties include `ws.properties` and `myres.properties`.

8.1 ws.properties

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

Table 8: ws.properties entries

`com.sybase.ase.ws.producer.httpport`
Indicates the port on which the SAP ASE Web Services Engine should listen for an HTTP connection. The default entry is `8181`.
`com.sybase.ase.ws.consumer.cisport`
Indicates the port on which the SAP ASE Web Services Engine should listen for TDS. The default entry is `8183`.
`com.sybase.ase.ws.logfilename`
Indicates where the log file for the SAP ASE Web Services Engine should be placed. The default location for UNIX is `$SYBASE/WS-16_0/logs/webservice.log`, or `%SYBASE%\WS-16_0\logs\webservice.log` for Windows.
`com.sybase.ase.ws.producer.jettylogfile`
Indicates where the log file for HTTP requests should be placed. The default location is for UNIX is `$SYBASE/WS-16_0/logs/http.log`, or `%SYBASE%\WS-16_0\logs\http.log` for Windows.
`com.sybase.ase.ws.producer.jettylogRetainDays`
Indicates the number of days the log is retained. The default is 90
`com.sybase.ase.ws.producer.jettylogAppend`
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 SAP ASE. 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.ssl.exciphersuites`

Specifies the weak cipher suites to disable to prevent SSL attacks and browser errors. The default RFC values are: `TLS_DHE_RSA_WITH_AES_128_CBC_SHA256`,`TLS_DHE_RSA_WITH_AES_128_CBC_SHA`.

`com.sybase.ase.ws.producer.ssl.exprotocols`

Specifies the TLS protocol versions to disable to prevent SSL attacks and browser errors. The default is `SSLv2Hello`,`SSLv3`.

`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`.

`com.sybase.ase.ws.producer.ssl.keypassword`

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

`com.sybase.ase.ws.producer.ssl.keystore`

Indicates the location of the keystore for SSL. The default location for UNIX is `$SYBASE/WS-16_0/producer/keystore`, or `%SYBASE%\WS-16_0\producer\keystore` for Windows.

`com.sybase.ase.ws.producer.ssl.httpsport`

Indicates the port on which the SAP ASE Web Services Engine should listen for an HTTPS connection. The default entry is `8182`.

`com.sybase.ase.ws.producer.ssl.password`

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

8.2   myres.properties

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

Table 9: myres.properties entries

Web Services – Set the following entries to configure Web Services.

`ws.ini`

Indicates the location of the `interfaces` or `sql.ini` file for SAP ASE. The default location for UNIX is `$SYBASE/interfaces`, or `%SYBASE%\ini\sql.ini` for Windows.

`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.

`ws.producer.port`

Indicates the port for the SAP ASE Web Services Engine.

`ws.producer.log`

Indicates the location of the `producer.log` file.

`ws.producer.jettylogfile`

Indicates where the log file for HTTP requests should be placed. The default location is for UNIX is `$SYBASE/WS-16_0/logs/http.log`, or `%SYBASE%\WS-16_0\logs\http.log` 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-16_0/producer/keystore`, or `%SYBASE%\WS-16_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 SAP ASE Web Services Engine as specified in the `interfaces` or `sql.ini` file.

`ws.consumer.host`

Indicates the host machine of the SAP ASE Web Services Engine.

`ws.consumer.port`

Indicates the port number for the SAP 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 SAP ASE server on which to run `installws`.

`ws.sqlsrv.sa_login`

Indicates the user login for the SAP ASE server.

`ws.sqlsrv.sa_password`

Indicates the password for the SAP ASE server.

8.3 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
```

9 SOAP and SAP ASE Datatype Mapping

Datatype mappings between SOAP and SAP ASE are described with respect to the Web Services feature.

9.1 SOAP-to-SAP ASE Datatype Mappings

SOAP datatypes have corresponding types in SAP ASE. These are used when mapping RPC/encoded Web services to proxy tables in SAP ASE.

SOAP Datatype	SAP ASE Datatype
string	varchar – 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
gYear	datetime
gMonthDay	datetime
gDay	datetime
gMonth	datetime
hexBinary	Unsupported
base64Binary	Unsupported
anyURI	varchar – length depends on Adaptive Server Enterprise page size

SOAP Datatype	SAP ASE Datatype
QName	varchar – length depends on Adaptive Server Enterprise page size
NOTATION	varchar – length depends on Adaptive Server Enterprise page size
normalizedString	varchar – length depends on Adaptive Server Enterprise page size
token	varchar – length depends on Adaptive Server Enterprise page size
language	varchar – length depends on Adaptive Server Enterprise page size
NMTOKEN	varchar – length depends on Adaptive Server Enterprise page size
Name	varchar – length depends on Adaptive Server Enterprise page size
NCName	varchar – length depends on Adaptive Server Enterprise page size
ID	varchar – length depends on Adaptive Server Enterprise page size
IDREF	varchar – length depends on Adaptive Server Enterprise page size
ENTITY	varchar – 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 Datatype	SAP ASE Datatype
`soap` complex datatypes	Not supported

9.2   SAP ASE-to-SOAP Datatype Mappings for the create service Command

There are relationship mappings between datatypes available as a type to a stored procedure and the datatype used for a SOAP user-defined Web service.

> **i 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`.

SAP 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`

SAP ASE Datatype	SOAP Parameter Type
char(n)	xsd:string
varchar(n)	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

10   Glossary

This glossary includes terms that are specific to the SAP ASE Web Services Engine.

SAP ASE Web Services Engine	The component of SAP ASE Web Services that acts as a producer and consumer of Web services. The SAP ASE Web Services Engine enables a client application to access SAP ASE stored procedures and SQL as Web methods. The SAP 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.
document/literal	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.
DTD	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.
HTTP	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.
HTTPS	HTTP with SSL or TLS encryption.
LDAP	Lightweight Directory Access Protocol, an application-level protocol for Internet-based directory services.
RPC/encoded	A type of Web method invoked with SOAP messages containing an XML element for each method parameter.
schema	An outline defining the structure, content, and semantics of an XML document.
SDS	Specialty Data Store, used as a Component Integration Service (CIS) to map a Web method to a proxy table.
SOAP	Simple Object Access Protocol, a standard for XML-based messaging across a network.
SQLX	SQL/XML, an ANSI and ISO standard that provides support for using XML in the context of a SQL database system.
UDDI	Universal Description Discovery and Integration.
URI	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).
URL	Uniform Resource Locator, one type of URI.
User-defined Web service	An arbitrary SQL statement named by the end user and executed with a SOAP client or through a Web browser.

W3C	World Wide Web Consortium, which produces the software standards for the World Wide Web.
Web method	A function described by WSDL and invoked through SOAP message.
Web service	One or more Web methods described by a WSDL file.
WSDL	Web Services Description Language, which describes the public interface to a Web service.
Xerces	The Apache open-source XML parser.
XML	Extensible Markup Language, a markup language standardized by W3C.
XML schema	A description of an XML document consisting of structure and content constraints.
XPath	XML Path Language, a language for addressing parts of an XML document.
XQL	XML Query Language, a precursor of XQuery.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon ![icon] : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:

    - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
    - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.

- Links with the icon ![icon]: You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

THE BEST RUN SAP