PROGRESS OpenEdge

White Paper: Addressing the POODLE Security Vulnerability in Progress OpenEdge

PROGRESS

Notices

© 2015 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

Business Making Progress, Corticon, DataDirect (and design), DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, Deliver More Than Expected, Easyl, Fathom, Icenium, Kendo UI, Making Software Work Together, OpenEdge, Powered by Progress, Progress, Progress Control Tower, Progress RPM, Progress Software Business Making Progress, Progress Software Developers Network, Rollbase, RulesCloud, RulesWorld, SequeLink, SpeedScript, Stylus Studio, TeamPulse, Telerik, Test Studio, and WebSpeed are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. AccelEvent, AppsAlive, AppServer, BravePoint, BusinessEdge, DataDirect Spy, DataDirect SupportLink, , Future Proof, High Performance Integration, Modulus, NativeScript, OpenAccess, Pacific, ProDataSet, Progress Arcade, Progress Pacific, Progress Profiles, Progress Results, Progress RFID, Progress Progress Software, ProVision, PSE Pro, SectorAlliance, Sitefinity, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, WebClient, and Who Makes Progress are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

Please refer to the Release Notes applicable to the particular Progress product release for any third-party acknowledgements required to be provided in the documentation associated with the Progress product.

The Release Notes can be found in the OpenEdge installation directory and online at:
https://community.progress.com/technicalusers/w/openedgegeneral/1329.openedge-product-documentation-overview.aspx.

For the latest documentation updates see OpenEdge Product Documentation on Progress Communities: (https://community.progress.com/technicalusers/w/openedgegeneral/1329.openedge-product-documentation-overview.aspx).

May 2015

Last updated with new content: Release 11.5.1

Table of Contents

Chapter 1: Introduction.....7

Chapter 2: Mitigating POODLE SSL 3.0 vulnerability in Progress OpenEdge 9

Cryptographic protocols and ciphers for Progress OpenEdge clients 9
Cryptographic protocols and ciphers for Progress OpenEdge servers 12

Chapter 3: Impact of POODLE vulnerability mitigation.....15

SSL communication compatibility matrix.....16
Testing for the POODLE SSL 3.0 vulnerability.....18
Downgrading security for SSL.....18
Changing the default protocols and ciphers.....18
Changing the default protocols and ciphers for Progress OpenEdge clients 18
Changing the default protocols and ciphers for Progress OpenEdge servers.....23

1

Introduction

Padding Oracle On Downgraded Legacy Encryption (POODLE) is a vulnerability that was identified in late 2014 and can affect secure communications making use of the Secure Socket Layer (SSL) 3.0 or earlier protocol. However, newer releases of secure communication protocols, including Transport Layer Security (TLS) 1.0, TLS 1.1 and TLS 1.2, are increasingly less susceptible. OpenEdge applications can be vulnerable to POODLE attacks due to the way the secure communication was established prior to 11.5.1 by giving preference to SSL 3.0 over TLS 1.0. Progress OpenEdge now establishes secure communications over TLS 1.0 by default instead of SSL 3.0.

POODLE allows an attacker to decrypt cipher text using a padding oracle side-channel attack. Applications that use cipher-block chaining with the SSL 3.0 communication protocol are vulnerable to POODLE attacks. The SSL 3.0 protocol does not adequately check the padding bytes that are sent with encrypted messages. These padding bytes can be replaced by the attacker. So, an attacker can exploit this vulnerability to decrypt and extract information from inside an encrypted transaction.

Note: For more information on the encryption flaws of SSL 3.0, see https://www.openssl.org/~bodo/ssl-poodle.pdf.

2

Mitigating POODLE SSL 3.0 vulnerability in Progress OpenEdge

Progress OpenEdge 11.5.1 onwards, all clients and servers are protected from the POODLE SSL 3.0 vulnerability. This has been done by replacing SSL 3.0 with TLS version 1.0 as the default SSL protocol. For secure communication, Progress OpenEdge recommends that you use TLS version 1.0 as the SSL protocol for all your Progress OpenEdge clients and servers with any cipher that is TLS version 1.0 compliant.

Note: Progress OpenEdge 11.5.1 uses OpenSSL Library 0.9.8g. This protects all Progress OpenEdge products against SSL 3.0 vulnerabilities. For more information about known vulnerabilities of OpenSSL and the release notes, see https://www.openssl.org/news/openssl-0.9.8-nos.html .

Cryptographic protocols and ciphers for Progress OpenEdge clients

The following table lists all the supported protocols and ciphers for Progress OpenEdge clients. When you install OpenEdge, all the default protocols are used, you can change the default to the other supported protocol (as listed in the table).

Note: All the ciphers are supported by default. You can use either the short name or the long name of the ciphers.

Component	Cryptographic Protocols	Supported ciphers	
		Short form	Long form

ABL Client	`TLSv1` (Default)
	Supported:
	`SSLv3`
		`AES128-SHA`
		`RC4-SHA`
		`RC4-MD5`
		`DES-CBC3-SHA`
		`DES-CBC-SHA`
		`EXP-DES-CBC-SHA`
		`EXP-RC4-MD5`
			`TLS_RSA_WITH_AES_128_CBC_SHA`
			`SSL_RSA_WITH_RC4_128_SHA`
			`SSL_RSA_WITH_RC4_128_MD5`
			`SSL_RSA_WITH_3DES_EDE_CBC_SHA`
			`SSL_RSA_WITH_DES_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_RC4_40_MD5`

ABL Client (as an HTTP client)
ABL Client (connecting to an external Web service)
	`TLSv1` (Default)
	Supported:
	`SSLv3`
		`AES128-SHA`
		`RC4-SHA`
		`RC4-MD5`
		`DES-CBC3-SHA`
		`DES-CBC-SHA`
		`EXP-DES-CBC-SHA`
		`EXP-RC4-MD5`
			`TLS_RSA_WITH_AES_128_CBC_SHA`
			`SSL_RSA_WITH_RC4_128_SHA`
			`SSL_RSA_WITH_RC4_128_MD5`
			`SSL_RSA_WITH_3DES_EDE_CBC_SHA`
			`SSL_RSA_WITH_DES_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_RC4_40_MD5`

OpenEdge WSDL Analyzer	`TLSv1` (Default)
	Supported:
	`SSLv3`
		`AES128-SHA`
		`RC4-SHA`
		`RC4-MD5`
		`DES-CBC3-SHA`
		`DES-CBC-SHA`
		`EXP-DES-CBC-SHA`
		`EXP-RC4-MD5`
			`TLS_RSA_WITH_AES_128_CBC_SHA`
			`SSL_RSA_WITH_RC4_128_SHA`
			`SSL_RSA_WITH_RC4_128_MD5`
			`SSL_RSA_WITH_3DES_EDE_CBC_SHA`
			`SSL_RSA_WITH_DES_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_RC4_40_MD5`

Java Open Client	`TLSv1` (Default)
	Supported:
	`SSLv3`
		`AES128-SHA`
		`RC4-SHA`
		`RC4-MD5`
		`DES-CBC3-SHA`
		`DES-CBC-SHA`
		`EXP-DES-CBC-SHA`
		`EXP-RC4-MD5`
			`TLS_RSA_WITH_AES_128_CBC_SHA`
			`SSL_RSA_WITH_RC4_128_SHA`
			`SSL_RSA_WITH_RC4_128_MD5`
			`SSL_RSA_WITH_3DES_EDE_CBC_SHA`
			`SSL_RSA_WITH_DES_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_RC4_40_MD5`

Component	Cryptographic Protocols	Supported ciphers	
		Short form	Long form
.NET Open Client	`TLSv1` (Default) Supported: `SSLv3`	All the ciphers that .NET framework 4.0 supports	

AppServer Internet Adapter (AIA)	`TLSv1` (Default)
	Supported:
	`SSLv3`
		`AES128-SHA`
		`RC4-SHA`
		`RC4-MD5`
		`DES-CBC3-SHA`
		`DES-CBC-SHA`
		`EXP-DES-CBC-SHA`
		`EXP-RC4-MD5`
			`TLS_RSA_WITH_AES_128_CBC_SHA`
			`SSL_RSA_WITH_RC4_128_SHA`
			`SSL_RSA_WITH_RC4_128_MD5`
			`SSL_RSA_WITH_3DES_EDE_CBC_SHA`
			`SSL_RSA_WITH_DES_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_RC4_40_MD5`

Web Services Adapter (WSA)	`TLSv1` (Default)
	Supported:
	`SSLv3`
		`AES128-SHA`
		`RC4-SHA`
		`RC4-MD5`
		`DES-CBC3-SHA`
		`DES-CBC-SHA`
		`EXP-DES-CBC-SHA`
		`EXP-RC4-MD5`
			`TLS_RSA_WITH_AES_128_CBC_SHA`
			`SSL_RSA_WITH_RC4_128_SHA`
			`SSL_RSA_WITH_RC4_128_MD5`
			`SSL_RSA_WITH_3DES_EDE_CBC_SHA`
			`SSL_RSA_WITH_DES_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_RC4_40_MD5`

WebSpeed Messenger	`TLSv1` (Default)
	Supported:
	`SSLv3`
		`AES128-SHA`
		`RC4-SHA`
		`RC4-MD5`
		`DES-CBC3-SHA`
		`DES-CBC-SHA`
		`EXP-DES-CBC-SHA`
		`EXP-RC4-MD5`
			`TLS_RSA_WITH_AES_128_CBC_SHA`
			`SSL_RSA_WITH_RC4_128_SHA`
			`SSL_RSA_WITH_RC4_128_MD5`
			`SSL_RSA_WITH_3DES_EDE_CBC_SHA`
			`SSL_RSA_WITH_DES_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_RC4_40_MD5`

Component	Cryptographic Protocols	Supported ciphers	
		Short form	Long form

REST	`TLSv1` (Default)
	Supported:
	`SSLv3`
		`AES128-SHA`
		`RC4-SHA`
		`RC4-MD5`
		`DES-CBC3-SHA`
		`DES-CBC-SHA`
		`EXP-DES-CBC-SHA`
		`EXP-RC4-MD5`
			`TLS_RSA_WITH_AES_128_CBC_SHA`
			`SSL_RSA_WITH_RC4_128_SHA`
			`SSL_RSA_WITH_RC4_128_MD5`
			`SSL_RSA_WITH_3DES_EDE_CBC_SHA`
			`SSL_RSA_WITH_DES_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_RC4_40_MD5`

SonicESB Adapter	`TLSv1` (Default)
	Supported:
	`SSLv3`
		`AES128-SHA`
		`RC4-SHA`
		`RC4-MD5`
		`DES-CBC3-SHA`
		`DES-CBC-SHA`
		`EXP-DES-CBC-SHA`
		`EXP-RC4-MD5`
			`TLS_RSA_WITH_AES_128_CBC_SHA`
			`SSL_RSA_WITH_RC4_128_SHA`
			`SSL_RSA_WITH_RC4_128_MD5`
			`SSL_RSA_WITH_3DES_EDE_CBC_SHA`
			`SSL_RSA_WITH_DES_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_RC4_40_MD5`

OE Client	`TLSv1.2` (Default) Supported: `SSLv3`	`AES128-SHA`	`TLS_RSA_WITH_AES_128_CBC_SHA`
OEM/OEE (using ActiveMQ Client)	`TLSv1.2` (Default) Supported: `TLSv1.1`, `TLSv1`, and `SSLv3`	All the cipher suites that are provided by the SSL implementation of the Java Secure Socket Extension (JSSE) are supported	

Cryptographic protocols and ciphers for Progress OpenEdge servers

The following table lists all the supported protocols and ciphers for Progress OpenEdge servers. When you install OpenEdge, all the default protocols are used, you can change the default to the other supported protocol (as listed in the table).

Note: You can use either the short name or the long name of the ciphers.

Component	Cryptographic Protocols	Supported ciphers	
		Short form	Long form

Broker	`TLSv1` (Default)
	Supported:
	`SSLv3`
		`AES128-SHA` (Default)
		Supported:
		`RC4-SHA`
		`RC4-MD5`
		`DES-CBC3-SHA`
		`DES-CBC-SHA`
		`EXP-DES-CBC-SHA`
		`EXP-RC4-MD5`
			`TLS_RSA_WITH_AES_128_CBC_SHA` (Default)
			Supported:
			`SSL_RSA_WITH_RC4_128_SHA`
			`SSL_RSA_WITH_RC4_128_MD5`
			`SSL_RSA_WITH_3DES_EDE_CBC_SHA`
			`SSL_RSA_WITH_DES_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_RC4_40_MD5`

Agent	`TLSv1` (Default)
	Supported:
	`SSLv3`
		`AES128-SHA` (Default)
		Supported:
		`RC4-SHA`
		`RC4-MD5`
		`DES-CBC3-SHA`
		`DES-CBC-SHA`
		`EXP-DES-CBC-SHA`
		`EXP-RC4-MD5`
			`TLS_RSA_WITH_AES_128_CBC_SHA` (Default)
			Supported:
			`SSL_RSA_WITH_RC4_128_SHA`
			`SSL_RSA_WITH_RC4_128_MD5`
			`SSL_RSA_WITH_3DES_EDE_CBC_SHA`
			`SSL_RSA_WITH_DES_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
			`SSL_RSA_EXPORT_WITH_RC4_40_MD5`

MS SQL server	`TLSv1` (Default)
	Supported:
	`SSLv3`
		`RC4-SHA`
		`RC4-MD5`
		(Default)
			`SSL_RSA_WITH_RC4_128_SHA`
			`SSL_RSA_WITH_RC4_128_MD5`

OpenEdge RDBMS	`TLSv1` (Default) Supported: `SSLv3`	`AES128-SHA` (Default)	`TLS_RSA_WITH_AES_128_CBC_SHA` (Default)

Component	Cryptographic Protocols	Supported ciphers	
		Short form	Long form

Pacific Application Server for OpenEdge (PAS for OpenEdge)	`TLSv1`, `TLSv1.1`, and `TLSv1.2` (Default)
	Supported:
	`SSLv3`
	`SSLv2`
		All the cipher suites that are provided by the SSL implementation of the JSSE are supported	

OEM/OEE (Remote AdminServer using ActiveMQ Server)	`TLSv1.2` (Default) Supported: `TLSv1.1`, `TLSv1`, and `SSLv3`		
OEM/OEE (WebServer)	`TLSv1.2` (Default) Supported: `TLSv1.1`, `TLSv1`, and `SSLv3`		
OEM/OEE (Remote trending database)	`TLSv1`, `TLSv1.1`, and `TLSv1.2` (Default) Supported: `SSLv3`		
OEM/OEE (Email alerts configuration)	`TLSv1`, `TLSv1.1`, and `TLSv1.2` (Default) Supported: `SSLv3`		
OEM/OEE (PAS for OpenEdge)	`TLSv1`, `TLSv1.1`, and `TLSv1.2` (Default) Supported: `SSLv3`		

The following OpenEdge server components do not support the SSL 3.0 or TLS protocols:

- OpenEdge Replication
- DataServers

3

Impact of POODLE vulnerability mitigation

As part of the POODLE SSL 3.0 vulnerability mitigation, the following is the impact on all Progress OpenEdge client and server components:

- **Backwards compatibility with SSL 3.0** –TLS version 1.0 is the default SSL protocols for all Progress OpenEdge 11.5.1 client and server components. To ensure interoperability with other components configured to use only SSL 3.0, the Progress OpenEdge 11.5.1 components are backwards compatible with SSL 3.0.
- **Performance degradation** – In Progress OpenEdge 11.5.1, the default SSL cipher suites have been changed to those that have fewer vulnerabilities. As a tradeoff for more security, you may find degradation in system performance.

SSL communication compatibility matrix

The following table shows the compatibility between the default SSL protocol configuration of 11.4 client components and 11.5.1 server components:

11.4 client component	11.5.1 server component	Is the server downgrade to SSL 3.0 required?
ABL Client	AppServer	NO
	Database Server	NO
ABL Client (connecting to an external Web service)	Web service	NO
Java Open Client	AppServer	YES
.NET Open Client	AppServer	YES
AppServer Internet Adapter (AIA) (Both HTTPS and SSL clients)	AppServer	YES
Web Services Adapter (WSA) (Both HTTPS and SSL clients)	AppServer	YES
REST (Both HTTPS and SSL clients)	AppServer	YES
Sonic ESB Adapter	AppServer	YES
SonicMQ Adapter	AppServer	NO
WebSpeed CGIIP Messenger (Both HTTPS and SSL clients)	WebSpeed Broker	YES
WebClient	AppServer-hosted application	NO
	Application connected to an AppServer	NO
	Application connected to an AppServer using AIA	NO

Note: On Windows XP systems, a pre-11.5.1 .NET Open Client cannot communicate with an 11.5.1 AppServer that is configured to use SSL 3.0. For such communication to occur, you must upgrade the Windows version.

The following table shows the compatibility between the default SSL protocol configuration of 11.5.1 client components and 11.4 server components:

11.5.1 client component	11.4 server component	Is the server downgrade to SSL 3.0 required?
ABL Client	AppServer	NO
	Database Server	NO
ABL Client (connecting to an external Web service)	Web service	NO
Java Open Client	AppServer	NO
.NET Open Client	AppServer	NO
AIA (Both HTTPS and SSL clients)	AppServer	NO
WSA (Both HTTPS and SSL clients)	AppServer	NO
REST (Both HTTPS and SSL clients)	AppServer	NO
Sonic ESB Adapter	AppServer	NO
SonicMQ Adapter	AppServer	NO
WebSpeed CGIIP Messenger (Both HTTPS and SSL clients)	WebSpeed Broker	NO
WebClient	AppServer-hosted application	NO
	Application connected to an AppServer	NO
	Application connected to an AppServer using AIA	NO
OpenEdge Management (using ActiveMQ Client)	Remote AdminServer (using ActiveMQ Server)	NO
OpenEdge BP Server	OpenEdge Database	NO

The following table shows the compatibility between the default SSL protocol configuration of 11.4 client components and 11.5.1 client components on PAS for OpenEdge:

11.4 client component	11.5.1 server component	Is the server downgrade to SSL 3.0 required?
ABL Client	PAS for OpenEdge	YES
Java Open Client	PAS for OpenEdge	YES

Testing for the POODLE SSL 3.0 vulnerability

Using the SSLC command line client (available at `$DLC/bin`), you can determine whether a client can connect to a POODLE SSL 3.0 safe server:

```
sslc s_client -connect host_name:port -tls1
```

If the handshake between the client and server is successful, the server is POODLE SSL 3.0 safe.

Downgrading security for SSL

Progress OpenEdge release 11.5.1 provides configuration parameters for changing the default protocols and ciphers across all client and server components. For certain client or server components, you may want to continue using cryptographic protocols or ciphers that are not POODLE SSL 3.0 vulnerability safe until those components are upgraded. In this case, you can selectively downgrade the security of the communication to support such non-compliant components. Progress does not recommend downgrading the protocol to SSL 3.0.

Note: A secure communication occurs only if both client and server support the same protocol and cipher.

Changing the default protocols and ciphers

Progress OpenEdge provides a set of configuration properties for you to change the default cryptographic protocols and ciphers across all OpenEdge client and server components.

Note: If you need to debug the SSL communication, you can set the `SSLSYS_DEBUG_LOGGING` environment variable from `1` to `5` (`5` fetches the maximum details) in the OS system variable or in the `Environment` section of the `ubroker.properties` file. This generates the `cert.server.log` and `cert.client.log` files.

Changing the default protocols and ciphers for Progress OpenEdge clients

You can set the protocols and the ciphers for the following client components:

- **ABL Client**:
  - To set the protocol and cipher for all ABL client connections, export the following environment variables in the startup script of the client application:

Property	Description
PSC_SSLCLIENT_PROTOCOLS	Accepts a comma-separated list of cryptographic protocols that are set for the client
PSC_SSLCLIENT_CIPHERS	Accepts a comma-separated list of valid cryptographic ciphers that are set for the client

Once you export the variables, invoke a Progress OpenEdge client executable and start a session for the client application to use the set cryptographic protocols and ciphers.

- To set the protocols and ciphers for single client connection, you can set the following parameters in the connection-parameters argument of the client's CONNECT() method:

Connection parameter	Description
-sslProtocols *SSL-protocol-names*	Accepts a comma-separated list of cryptographic protocols that are set for the client
-sslCiphers *SSL-protocol-ciphers*	Accepts a comma-separated list of valid cryptographic ciphers that are set for the client

Note: You can use ABL clients to create connections with different server components. If you do not set the -sslprotocols and -sslciphers connection parameters using the client's CONNECT() method, the protocol and cipher values are set using the PSC_SSLCLIENT_PROTOCOLS and PSC_SSLCLIENT_CIPHERS environment variables in the ubroker.properties file (available at OpenEdge-install-directory/properties).

- **OpenEdge WSDL Analyzer**: To set the protocol and cipher for OpenEdge WSDL Analyzer, export the following environment variables in the startup script of the client application:

Property	Description
PSC_SSLCLIENT_PROTOCOLS	Accepts a comma-separated list of cryptographic protocols
PSC_SSLCLIENT_CIPHERS	Accepts a comma-separated list of valid cryptographic ciphers

Once you export the variables, invoke the bprowsdldoc client executable to use the set cryptographic protocols and ciphers.

- **Java Open Client**: In the client's Connection object, you can set the following properties using the setStringProperty method of the RunTimeProperties class:

Property	Description
PROGRESS.Session.sslProtocols	Accepts a comma-separated list of cryptographic protocols that are set for the client
PROGRESS.Session.sslCiphers	Accepts a comma-separated list of valid cryptographic ciphers that are set for the client

You can also set these properties as Java system properties.

- **.NET Open Client**: In the client's app.config file, you can set the Progress.Open4GL.RunTimeProperties.SSLProtocols runtime property with a comma-separated list of cryptographic protocols. The cryptographic ciphers are automatically set using the .NET framework.
- **AppServer Agent**: In the [Environment.<*appserver_broker*>] section of the ubroker.properties file (available at OpenEdge-install-directory/properties), you can set the following environment variables:

Property	Description
PSC_SSLCLIENT_PROTOCOLS	Accepts a comma-separated list of cryptographic protocols that are set for the agent
PSC_SSLCLIENT_CIPHERS	Accepts a comma-separated list of valid cryptographic ciphers that are set for the agent

Note: Even if a secure connection is established, a known AIA logging error causes the Error Setting SSL parameters from ServerConnection error statement to appear in the log file. To avoid this problem, set the PSC_SSLCLIENT_PROTOCOLS and PSC_SSLCLIENT_CIPHERS properties in the Apache Tomcat catalina shell script file.

- **WebSpeed Agent**: In the [Environment.<*webspeed_broker*>] section of the ubroker.properties file (available at OpenEdge-install-directory/properties), you can set the following environment variables:

Property	Description
PSC_SSLCLIENT_PROTOCOLS	Accepts a comma-separated list of cryptographic protocols that are set for the agent
PSC_SSLCLIENT_CIPHERS	Accepts a comma-separated list of valid cryptographic ciphers that are set for the agent

- **AppServer Internet Adapter**: In the startup script of the client application, export the following environment variables:

Property	Description
`PSC_SSLCLIENT_PROTOCOLS`	Accepts a comma-separated list of cryptographic protocols that are set for the AIA instance
`PSC_SSLCLIENT_CIPHERS`	Accepts a comma-separated list of valid cryptographic ciphers that are set for the AIA instance

After you export the variables, invoke a Progress OpenEdge client executable and start a session for the client application to use the set protocols and ciphers.

- **WebSpeed Messenger**: In the startup script of the client application, export the following environment variables:

Property	Description
`PSC_SSLCLIENT_PROTOCOLS`	Accepts a comma-separated list of cryptographic protocols that are set for the WebSpeed client instance
`PSC_SSLCLIENT_CIPHERS`	Accepts a comma-separated list of valid cryptographic ciphers that are set for the WebSpeed client instance

After you export the variables, invoke an Progress OpenEdge client executable and start a session for the client application to use the set protocols and ciphers.

- **SonicESB Adapter**: Using the **Sonic Management Console**, you can set the following runtime properties:

Property	Description
`sslProtocols`	Accepts a comma-separated list of cryptographic protocols that are set for the SonicESB adapter
`sslCiphers`	Accepts a comma-separated list of valid cryptographic ciphers that are set for the SonicESB adapter

To set these properties:

1. On the **Sonic Management Console**, select the **Configure** tab.
2. Under **Services**, select **OpenEdge Native Services**.
3. Select the **dev.OpenEdge** service name.
4. Under **Init Parameters**, click **Runtime Properties**. A list of properties and their current values is displayed.
5. For the **sslProtocols** and **sslCiphers** properties, enter comma-separated values of cryptographic protocols and ciphers that you want to set for the adapter.

- **Web Services Adapter**: In the `default.props` file (available at the `OpenEdge-install-directory/webapps/wsa/wsa1`), set the following properties:

Property	Description
`sslProtocols`	Accepts a comma-separated list of cryptographic protocols that are set for the WSA application
`sslCiphers`	Accepts a comma-separated list of valid cryptographic ciphers that are set for the WSA application

To set the default protocol and cipher values for a WSA client, set the properties as `Default`. The following code shows how to set the properties.

```
<?xml version="1.0" encoding="UTF-8"?>
<ApplicationRuntimeProperties xmlns="urn:schemas-progress-com:WSAD:0011"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
...
<sslCiphers>Default</sslCiphers>
<sslProtocols>Default</sslProtocols>
...
</ApplicationRuntimeProperties>
```

- **REST Adapter**: In the `runtime.props` file (available at `$CATALINA_HOME/webapps/application_name/WEB-INF/adapters`), set the following properties:

Property	Description
`sslProtocols`	Accepts a comma-separated list of cryptographic protocols that are set for the REST application
`sslCiphers`	Accepts a comma-separated list of valid cryptographic ciphers that are set for the REST application

- **OpenEdge Management and OpenEdge Explorer (using ActiveMQ client)**: To set the cryptographic protocol and cipher for a remote AdminServer, you must set the following properties in the `management.properties` file that is available at `$DLC/properties`:

Property	Description
`sslEnable`	**Default value**: `0` Enables you to set the cryptographic protocols and ciphers for secure communication with a remote AdminServer. To set the default cryptographic protocols and ciphers for the remote AdminServer, set this property value to `1`. You must reset the remote AdminServer and OpenEdge Management or OpenEdge Explorer for the setting to take effect.
`enabledProtocols`	**Default value**: `TLSv1.2` **Supported values**: `SSLv3`, `TLSv1`, and `TLSv1.1` If you want to change the default cryptographic protocol for the remote AdminServer, enter this property in the `management.properties` file. The property accepts a comma-separated list of valid cryptographic protocols that are set for secure communication.
`enabledCipherSuites`	If you want to change the default cryptographic protocol for the remote AdminServer, enter this property in the `management.properties` file. The property accepts a comma-separated list of valid cryptographic protocols that are set for secure communication.

Changing the default protocols and ciphers for Progress OpenEdge servers

You can set the protocol and the cipher for the following server components:

- **AppServer Broker**: In the `[Environment.<appserver_broker>]` section of the `ubroker.properties` file, you can set the following environment variables:

Property	Description
`PSC_SSLSERVER_PROTOCOLS`	Accepts a comma-separated list of cryptographic protocols that are set for the AppServer instance
`PSC_SSLSERVER_CIPHERS`	Accepts a comma-separated list of valid cryptographic ciphers that are set for the AppServer instance. The default value is `AES128-SHA`.

Once you make a change in the `ubroker.properties` file, you must restart the AppServer broker for the new values to take effect.

- **WebSpeed Broker**: In the `[Environment.<webspeed_broker>]` section of the `ubroker.properties` file, you can set the following environment variables:

Property	Description
`PSC_SSLSERVER_PROTOCOLS`	Accepts a comma-separated list of cryptographic protocols that are set for the Web server instance
`PSC_SSLSERVER_CIPHERS`	Accepts a comma-separated list of valid cryptographic ciphers that are set for the Web server instance. The default value is `AES128-SHA`.

Once you make a change in the `ubroker.properties` file, you must restart the WebSpeed broker for the new values to take effect.

- **OpenEdge Database Server**: In the startup script of the server application, export the following environment variables.

Property	Description
`PSC_SSLSERVER_PROTOCOLS`	Accepts a comma-separated list of cryptographic protocols that are set for the database server instance
`PSC_SSLSERVER_CIPHERS`	Accepts a comma-separated list of valid cryptographic ciphers that are set for the database server instance. The default value is `AES128-SHA`.

Once you export the variables, invoke a database server executable and start a session for the client application to use the set protocols and ciphers.

- **OpenEdge RDBMS**: In the `ubroker.properties` file (available at `OpenEdge-install-directory/properties`), you can set the following environment variables:

Property	Description
`PSC_SSLSERVER_PROTOCOLS`	Accepts the cryptographic protocol that is set for the instance
`PSC_SSLSERVER_CIPHERS`	Accepts the valid cryptographic cipher that is set for the instance. The default value is `AES128-SHA`.

- **MS SQL Server**: In the `ubroker.properties` file (available at `OpenEdge-install-directory/properties`), you can set the following environment variables:

Property	Description
PSC_SSLSERVER_PROTOCOLS	Accepts the cryptographic protocol that is set for the instance
PSC_SSLSERVER_CIPHERS	Accepts the valid cryptographic cipher that is set for the instance. The default value is AES128-SHA.

- **AppServer Agent**: In the [Environment.<*appserver_broker*>] section of the ubroker.properties file (available at OpenEdge-install-directory/properties), you can set the following environment variables:

Property	Description
PSC_SSLSERVER_PROTOCOLS	Accepts the cryptographic protocol that is set for the agent
PSC_SSLSERVER_CIPHERS	Accepts the valid cryptographic cipher that is set for the agent. The default value is AES128-SHA.

- **WebSpeed Agent**: In the [Environment.<*webspeed_broker*>] section of the ubroker.properties file (available at OpenEdge-install-directory/properties), you can set the following environment variables:

Property	Description
PSC_SSLSERVER_PROTOCOLS	Accepts the cryptographic protocol that is set for the agent
PSC_SSLSERVER_CIPHERS	Accepts the valid cryptographic cipher that is set for the agent. The default value is AES128-SHA.

- **PAS for OpenEdge**: Use the tcman.sh config command to set the following Java system properties:

Property	Description
psc.as.https.protocol	Accepts a comma-separated list of SSL protocols that are set for the PAS for OpenEdge instance. The default values are TLSv1, TLSv1.1, and TLSv1.2.
psc.as.https.ciphers	Accepts a comma-separated list of valid cryptographic ciphers that are set for the PAS for Progress OpenEdge instance. By default, all the cipher suites are supported.

These properties are saved in the `catalina.properties` file (available at `$CATALINA_BASE/conf`).

- **Tomcat-in-the-box**: In the `server.xml` file (available at `$CATALINA_BASE/conf`), set the following properties:

Property	Description
`sslProtocols`	Accepts a comma-separated list of cryptographic protocols that are set for the instance. The default values are `TLSv1`, `TLSv1.1`, and `TLSv1.2`.
`ciphers`	Accepts a comma-separated list of valid cryptographic ciphers that are set for the instance. By default, all the cipher suites are supported.

For client and server components that are controlled by Apache Tomcat server, you can optionally set the server environment variables in the `proset` shell script file (available at `OpenEdge-install-directory`).

- **OpenEdge Management and OpenEdge Explorer**:
  - **OpenEdge Management WebServer**: To set the cryptographic protocol and cipher for a WebServer, you must set the following properties in the `fathom.properties` file that is available at `$DLC/properties`:

Property	Description
`sslEnable`	**Default value**: `0` Enables you to change the cryptographic protocols and ciphers for secure communication with a WebServer. If you enable SSL for the WebServer in OpenEdge Management and OpenEdge Explorer, this property is set to `true`.

Property	Description
`SSLEnabledProtocols`	**Default value**: `TLSv1.2` **Supported values**: `SSLv3`, `TLSv1`, and `TLSv1.1` If you want to change the default cryptographic protocol for the WebServer, enter this property in the `fathom.properties` file. The property accepts a comma-separated list of valid cryptographic protocols that are set for secure communication.
`SSLEnabledCipherSuites`	If you want to change the default cryptographic ciphers for the WebServer, enter this property in the `fathom.properties` file. The property accepts a comma-separated list of valid cryptographic ciphers that are set for secure communication.

- **Remote trending database**: To set the cryptographic protocol for connecting to the remote trending database, do the following:
  1. From the management console menu bar, click **Options**. The **Options** page appears.
  2. Select **Trend Database**. The **OpenEdge Management Trend Database Configuration** page appears and displays the current remote trending database settings.
  3. Select **Store trend data in a remote OpenEdge Management instance**.
  4. To use HTTPS, select the **Use HTTPS (SSL) protocol** option.
  5. Click **Advanced Options**. The expanded **OpenEdge Management Trend Database Configuration** page appears.
  6. Under HTTPS Configuration, set the **Enabled protocols** options as required.
  7. Click **Save** to save your changes.

- **OpenEdge Management email alerts**: To set the cryptographic protocol for exchanging email alerts, do the following:
  1. From the management console menu bar, click **Options**. The **Options** page appears.
  2. Select **Email alerts**. The **OpenEdge Management Email Alerts Configuration** page appears and displays the current email alerts settings.
  3. Set the **Mail server (SMTP) SSL/TLS enabled protocols** options as required. You can also set the cipher suites by entering comma-separated values of supported cipher suites in the **Mail server(SMTP) SSL/TLS enabled cipher suites** field.
  4. Click **Save** to save your changes.

- **PAS for OpenEdge**: To set the cryptographic protocol for a PAS for OpenEdge instance, do the following:

1. Click **Resources** in the OpenEdge Management console menu. All resources managed by your console appear in the grid frame.
2. Filter, or search for, and select the required PAS for OpenEdge instance. For example, type the default PAS for OpenEdge instance name, `oepas1`, in the **Filter** field or select **Pacific Application Server** as the selection in the **Type** drop-down menu. The **Pacific Application Server: *<instance name>*** page appears.
3. In the **Command and control** section of the page, click **Configuration**. The **Property Configuration** page appears.
4. In the **Basic** tab, set the **SSL enabled protocols** options as required. You can also set the cipher suites by entering comma-separated values of supported cipher suites in the **SSL enabled cipher suites** field.
5. Click **Save** to save your changes. Any changes you make are also reflected in the `pasmgr.properties` file that is available at `$DLC/properties`.

Note: Both the AppServer Broker and Agent inherit environment variables (`PSC_SSLCLIENT_PROTOCOLS`, `PSC_SSLCLIENT_CIPHERS`, `PSC_SSLSERVER_PROTOCOLS`, and `PSC_SSLSERVER_CIPHERS`) both from the AdminServer process and the `[Environment.<broker_name>]` sections of the `ubroker.properties` file. The variables in the `[Environment.<broker_name>]` sections of the `ubroker.properties` file supercede the variables set by the AdminServer process. If you set these variables in the `ubroker.properties` file and then use the shell to manually start the AdminServer process, the brokers and agents inherit the variables set in the `ubroker.properties` file.