

# Camel Configuration Utilities

## JSSE Utility

The JSSE Utility, available as of **2.8**, allows you to easily configure aspects of the [Java Secure Socket Extension](#) (JSSE) API in order to greatly simplify the use of custom transport layer security (TLS) settings on Camel components.

### Supported Components

The following Camel components directly support the use of this configuration utility:

- [AHC](#)
- [AHC-WS](#)
- [APNS](#)
- [Box](#)
- [Cometd](#)
- [Consul](#)
- [Etc](#)
- [FTP, FTP2](#)
- [HTTP4](#)
- [IRC](#)
- [Jetty](#)
- [Kafka](#)
- [Lumberjack](#)
- [Mail](#)
- [MINA2](#)
- [NATS](#)
- [Netty, Netty4](#)
- [Olingo2](#)
- [Restlet](#)
- [Salesforce](#)
- [ServiceNow](#)
- [Stomp](#)
- [Undertow](#)
- [Websocket](#)

The following Camel components indirectly support the use of this configuration utility:

- [CXF](#)
- [HTTP](#)
- [REST Swagger Component](#)

### Configuration

The key component in configuring TLS through the JSSE API is the SSLContext. The [SSLContext](#) provides socket factories for both [client-side](#) and [server-side](#) sockets as well as another component called an [SSLEngine](#) that is used by non-blocking IO to support TLS. The JSSE configuration utility provides an easy to use builder for configuring these JSSE components, among others, in a manner that allows you to provide all configuration options up front during the initialization of your application such that you don't have to customize library code or dig through the inner workings of a third-party library in order to inject hooks for the configuration of each component in the JSSE API. The central builder in the JSSE configuration utility is the SSLContextParameters. This class serves as the entry point for most configuration in the JSSE utility.



All non-native classes are in the org.apache.camel.util.jsse package. All non-W3C schema defined types are in the <http://camel.apache.org/schema/spring> or <http://camel.apache.org/schema/blueprint> namespaces for Spring and Blueprint based configuration, respectively.

### SSLContextParameters

Java Field Name and Class	XML Attribute /Element and Type	Description
cipherSuites - CipherSuitesParameters	sslContextParameters /ciphersuites - CipherSuitesParameters	This optional property represents a collection of explicitly named cipher suites to enable on both the client and server side as well as in the SSLEngine. These values take precedence over filters supplied in cipherSuitesFilter. The utility attempts to enable the listed cipher suites regardless of whether or not the JSSE provider actually supports them or not. This behavior guarantees that listed cipher suites are always enabled when listed. For a more lenient option, use cipherSuitesFilter.

cipherSuitesFilter - <a href="#">FilterParameters</a>	sslContextParameters /cipherSuitesFilter - <a href="#">FilterParameters</a>	<p>This optional property represents a collection of include and exclude patterns for cipher suites to enable on both the client and server side as well as in the SSLEngine. The patterns are applied over only the available cipher suites. The exclude patterns have precedence over the include patterns. If no cipherSuites and no cipherSuitesFilter are present, the default patterns applied are:</p> <p>Includes</p> <ul style="list-style-type: none"> <li>• . *</li> </ul> <p>Excludes</p> <ul style="list-style-type: none"> <li>• . *_NULL_. *</li> <li>• . *_anon_. *</li> <li>• . *_DES_. * <b>Camel 2.15.4</b></li> <li>• . *_EXPORT_. * <b>Camel 2.15.4</b></li> </ul>
secureSocketProtocols - SecureSocketProtocolParameters	sslContextParameters /secureSocketProtocols - SecureSocketProtocolParameters	<p>This optional property represents a collection of explicitly named secure socket protocols, such as SSLv3/TLS/etc., to enable on both the client and server side as well as in the SSLEngine. These values take precedence over filters supplied in secureSocketProtocolsFilter. The utility attempts to enable the listed protocols regardless of whether or not the JSSE provider actually supports them or not. This behavior guarantees that listed protocols are always enabled when listed. For a more lenient option, use secureSocketProtocolsFilter.</p>
secureSocketProtocolsFilter - <a href="#">FilterParameters</a>	sslContextParameters /secureSocketProtocolsFilter - <a href="#">FilterParameters</a>	<p>This optional property represents a collection of include and exclude patterns for secure socket protocols to enable on both the client and server side as well as in the SSLEngine. The patterns are applied over only the available protocols. The exclude patterns have precedence over the include patterns. If no secureSocketProtocols and no secureSocketProtocolsFilter are present, the default patterns applied are:</p> <p>Includes</p> <ul style="list-style-type: none"> <li>• . *</li> </ul>
sessionTimeout - java.lang.String	sslContextParameters /@sessionTimeout - xsd:string	<p>This optional property defines the timeout period, in seconds, for sessions on both the client and server side as well as in the SSLEngine.</p>
keyManagers - <a href="#">KeyManagersParameters</a>	sslContextParameters /keyManagers - <a href="#">KeyManagersParameters</a>	<p>This optional property configures the source of key material for providing identity of client and server side connections as well as in the SSLEngine. If omitted, no source of key material is provided and the SSLContext is suitable only for client-side usage when mutual authentication is not in use. You typically configure this property with a key store containing a client or server private key.</p>
trustManagers - <a href="#">TrustManagersParameters</a>	sslContextParameters /trustManagers - <a href="#">TrustManagersParameters</a>	<p>This optional property configures the source of material for verifying trust of key material used in the handshake process. If omitted, the default trust manager is automatically used. See the <a href="#">JSSE documentation</a> for more information on how the default trust manager is configured. You typically configure this property with a key store containing trusted CA certificates.</p>
secureRandom - SecureRandomParameters	sslContextParameters /secureRandom - SecureRandomParameters	<p>This optional property configures the secure random number generator used by the client and server side as well as in the SSLEngine. If omitted, the default secure random number generator is used.</p>
clientParameters - <a href="#">SSLContextClientParameters</a>	sslContextParameters /clientParameters - <a href="#">SSLContextClientParameters</a>	<p>This optional property configures additional settings that apply only to the client side aspects of the SSLContext. If present, these settings override the settings specified at the SSLContextParameters level.</p>
serverParameters - <a href="#">SSLContextServerParameters</a>	sslContextParameters /serverParameters - <a href="#">SSLContextServerParameters</a>	<p>This optional property configures additional settings that apply only to the server side aspects of the SSLContext. If present, these settings override the settings specified at the SSLContextParameters level.</p>
provider - java.lang.String	sslContextParameters /@provider - xsd:string	<p>The optional provider identifier for the JSSE implementation to use when constructing the SSLContext. If omitted, the standard provider look-up mechanism is used to resolve the provider.</p>

secureSocketProtocol - java.lang.String	sslContextParameters /@secureSocketProtocol - xsd:string	The optional secure socket protocol. See <a href="#">Appendix A</a> in the Java Secure Socket Extension Reference Guide for information about standard protocol names. If omitted, TLS is used by default. Note that this property is related to but distinctly different from the secureSocketProtocols and secureSocketProtocolsFilter properties.
certAlias - java.lang.String	sslContextParameters /@certAlias - xsd:string	<b>Camel 2.13:</b> An optional certificate alias to use. This is useful when the keystore has multiple certificates.

## KeyManagersParameters


Java Field Name and Class	XML Attribute /Element and Type	Description
keyStore- <a href="#">KeyStoreParameters</a>	keyStore - <a href="#">KeyStoreParameters</a>	This optional property represents the key store that provides key material to the key manager. This is typically configured with a key store containing a user or server private key. In some cases, such as when using PKCS#11, the key store is omitted entirely.
keyPassword - java.lang.String	@keyPassword - xsd:string	The optional password for recovering/accessing the private key in the key store. This is typically the password for the private key in the configured key store; however, in some cases, such as when using PKCS#11, the key password may be provided through other means and is omitted entirely in this configuration.
provider - java.lang.String	@provider - xsd:string	The optional provider identifier for the KeyManagerFactory used to create the KeyManagers represented by this object's configuration. If omitted, the default look-up behavior is used.
algorithm - java.lang.String	@algorithm - xsd:string	The optional algorithm name for the KeyManagerFactory used to create the KeyManager represented by this object's configuration. See the <a href="#">Java Secure Socket Extension Reference Guide</a> for information about standard algorithm names.
trustManager - java.lang.String	@trustManager - xsd:string	<b>Camel 2.17:</b> To use a existing configured trust manager instead of using TrustManagerFactory to get the TrustManager.

## TrustManagersParameters

Java Field Name and Class	XML Attribute /Element and Type	Description
keyStore- <a href="#">KeyStoreParameters</a>	keyStore - <a href="#">KeyStoreParameters</a>	This optional property represents the key store that provides key material to the trust manager. This is typically configured with a key store containing trusted CA certificates / public keys. In some cases, such as when using PKCS#11, the key store is omitted entirely.
provider - java.lang.String	@provider - xsd:string	The optional provider identifier for the TrustManagerFactory used to create the TrustManagers represented by this object's configuration. If omitted, the default look-up behavior is used.
algorithm - java.lang.String	@algorithm - xsd:string	The optional algorithm name for the TrustManagerFactory used to create the TrustManager represented by this object's configuration. See the <a href="#">Java Secure Socket Extension Reference Guide</a> for information about standard algorithm names.

## KeyStoreParameters

Java Field Name and Class	XML Attribute /Element and Type	Description
---------------------------	---------------------------------	-------------

resource - java. lang. String	keyStore /@resource - xsd:string	<p>This optional property represents the location of the key store resource to load the key store from. In some cases, the resource is omitted as the key store content is provided by other means. The loading of the resource, if provided, is attempted by treating the resource as a file path, a class path resource, and a URL in that order. An exception is thrown if the resource cannot be resolved to readable input stream using any of the above methods.</p> <div>  <b>OSGi Usage</b>  For programmatic and Spring based XML configuration in OSGi, a resource specified as a classpath resource path may be accessible in the bundle containing the XML configuration file or in a package that is imported by that bundle. As Blueprint does not define the thread context classloader behavior, only classpath resources in the bundle containing the XML configuration file may be resolved from a Blueprint based XML configuration. </div>
password - java. lang. String	keyStore /@password - xsd:string	The optional password for reading/opening/verifying the key store.
type - java. lang. String	keyStore /@type - xsd:string	The optional type of the key store. See Appendix A in the <a href="#">Java Cryptography Architecture Standard Algorithm Name Documentation</a> for more information on standard names. If omitted, defaults to the default lookup mechanism as defined by <a href="#">KeyStore.getDefaultType()</a> .
provider - java. lang. String	keyStore /@provider - xsd:string	The optional provider identifier for the provider used to create the KeyStores represented by this object's configuration. If omitted, the default look-up behavior is used.

### FilterParameters

Java Field Name and Class	XML Attribute /Element and Type	Description
include - java.util. List<java.lang.String>	include - xsd:string	This optional property represents zero or more regular expression patterns for which matching values should be included. The list of excludes takes precedence over the include patterns.
exclude - java.util. List<java.lang.String>	exclude - xsd:string	This optional property represents zero or more regular expression patterns for which matching values should be included. The list of excludes takes precedence over the include patterns.

### SecureRandomParameters

Java Field Name and Class	XML Attribute /Element and Type	Description
algorithm - java. lang. String	@algorithm - xsd:string	This optional property represents the Random Number Generator (RNG) algorithm identifier for the SecureRandom factory method used to create the SecureRandom represented by this object's configuration. See <a href="#">Appendix A</a> in the Java Cryptography Architecture API Specification & Reference for information about standard RNG algorithm names.
provider - java. lang. String	@provider - xsd:string	The optional provider identifier for the SecureRandom factory method used to create the SecureRandom represented by this object's configuration. If omitted, the default look-up behavior is used.

### SSLContextServerParameters

Java Field Name and Class	XML Attribute /Element and Type	Description
cipherSuites - CipherSuitesParameters	sslContextClientParameters /ciphersuites - CipherSuitesParameters	This optional property represents a collection of explicitly named cipher suites to enable on the server side only (SSLServerSocketFactory/SSLServerSocket) by overriding the value of this setting in the SSLContextParameters. This option has no effect on the SSLContext configuration. These values take precedence over filters supplied in cipherSuitesFilter. The utility attempts to enable the listed cipher suites regardless of whether or not the JSSE provider actually supports them or not. This behavior guarantees that listed cipher suites are always enabled when listed. For a more lenient option, use cipherSuitesFilter.

cipherSuitesFilter - <a href="#">FilterParameters</a>	sslContextClientParameters /cipherSuitesFilter - <a href="#">FilterParameters</a>	This optional property represents a collection of include and exclude patterns for cipher suites to enable on the server side only (SSLServerSocketFactory/SSLServerSocket) by overriding the value of this setting in the SSLContextParameters. This option has no affect on the SSLEngine configuration. The patterns are applied over only the available cipher suites. The exclude patterns have precedence over the include patterns. See SSLContextParameters for details of the behavior if this option and cipherSuites is omitted at this level.
secureSocketProtocols - SecureSocketProtocolsParameters	sslContextClientParameters /secureSocketProtocols - SecureSocketProtocolsParameters	This optional property represents a collection of explicitly named secure socket protocols, such as SSLv3/TLS/etc., to enable on the server side only (SSLServerSocketFactory/SSLServerSocket) by overriding the value of this setting in the SSLContextParameters. This option has no affect on the SSLEngine configuration. These values take precedence over filters supplied in secureSocketProtocolsFilter. The utility attempts to enable the listed protocols regardless of whether or not the JSSE provider actually supports them or not. This behavior guarantees that listed protocols are always enabled when listed. For a more lenient option, use secureSocketProtocolsFilter.
secureSocketProtocolsFilter - <a href="#">FilterParameters</a>	sslContextClientParameters /secureSocketProtocolsFilter - <a href="#">FilterParameters</a>	This optional property represents a collection of include and exclude patterns for secure socket protocols to enable on the server side only (SSLServerSocketFactory/SSLServerSocket) by overriding the value of this setting in the SSLContextParameters. This option has no affect on the SSLEngine configuration. The patterns are applied over only the available protocols. The exclude patterns have precedence over the include patterns. See SSLContextParameters for details of the behavior if this option and/or secureSocketProtocols is omitted at this level.
sessionTimeout - java.lang.String	sslContextServerParameters /@sessionTimeout - xsd:string	This optional property defines the timeout period, in seconds, for sessions on the server side. This setting affects both the SSLServerSocketFactory/SSLServerSocket as well as the server side of the SSLEngine.
clientAuthentication - java.lang.String	sslContextServerParameters /@clientAuthentication - xsd:string	This optional property indicates if the server side does not request, requests, or requires clients to provide authentication credentials during the handshake process. This is commonly referred to as mutual authentication, two direction SSL/TLS, or two-legged SSL/TLS. Valid values are: NONE, WANT, REQUIRE

## SSLContextClientParameters

Java Field Name and Class	XML Attribute /Element and Type	Description
cipherSuites - CipherSuitesParameters	sslContextClientParameters /ciphersuites - CipherSuitesParameters	This optional property represents a collection of explicitly named cipher suites to enable on the client side only (SSLSocketFactory/SSLSocket) by overriding the value of this setting in the SSLContextParameters. This option has no affect on the SSLEngine configuration. These values take precedence over filters supplied in cipherSuitesFilter. The utility attempts to enable the listed cipher suites regardless of whether or not the JSSE provider actually supports them or not. This behavior guarantees that listed cipher suites are always enabled when listed. For a more lenient option, use cipherSuitesFilter.
cipherSuitesFilter - <a href="#">FilterParameters</a>	sslContextClientParameters /cipherSuitesFilter - <a href="#">FilterParameters</a>	This optional property represents a collection of include and exclude patterns for cipher suites to enable on the client side only (SSLSocketFactory/SSLSocket) by overriding the value of this setting in the SSLContextParameters. This option has no affect on the SSLEngine configuration. The patterns are applied over only the available cipher suites. The exclude patterns have precedence over the include patterns. See SSLContextParameters for details of the behavior if this option and cipherSuites is omitted at this level.
secureSocketProtocols - SecureSocketProtocolsParameters	sslContextClientParameters /secureSocketProtocols - SecureSocketProtocolsParameters	This optional property represents a collection of explicitly named secure socket protocols, such as SSLv3/TLS/etc., to enable on the client side only (SSLSocketFactory/SSLSocket) by overriding the value of this setting in the SSLContextParameters. This option has no affect on the SSLEngine configuration. These values take precedence over filters supplied in secureSocketProtocolsFilter. The utility attempts to enable the listed protocols regardless of whether or not the JSSE provider actually supports them or not. This behavior guarantees that listed protocols are always enabled when listed. For a more lenient option, use secureSocketProtocolsFilter.
secureSocketProtocolsFilter - <a href="#">FilterParameters</a>	sslContextClientParameters /secureSocketProtocolsFilter - <a href="#">FilterParameters</a>	This optional property represents a collection of include and exclude patterns for secure socket protocols to enable on the client side only (SSLSocketFactory/SSLSocket) by overriding the value of this setting in the SSLContextParameters. This option has no affect on the SSLEngine configuration. The patterns are applied over only the available protocols. The exclude patterns have precedence over the include patterns. See SSLContextParameters for details of the behavior if this option and/or secureSocketProtocols is omitted at this level.
sessionTimeout - java.lang.String	sslContextServerParameters /@sessionTimeout - xsd:string	This optional property defines the timeout period, in seconds, for sessions on the client side This setting affects both the SSLSocketFactory/SSLSocket as well as the client side of the SSLEngine.

sniHostName	sslContextClientParameters/sniHostNames	<b>Since 2.18.0.</b> You can use this optional property to set multiple sniHostName (xsd:string) elements to set the SNIHostNames to be used when communicating over TLS. For more information see <a href="https://en.wikipedia.org/wiki/Server_Name_Indication">https://en.wikipedia.org/wiki/Server_Name_Indication</a>
-------------	-----------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Examples

### Programmatic Usage

#### Setting Client Authentication On the Server Side

This configuration sets the server side aspects of the TLS configuration to require client authentication during the handshake process. This configuration uses the default trust store and a custom key store to provide key material for both the server and client sides of the SSLContext.

```
KeyStoreParameters ksp = new KeyStoreParameters();
ksp.setResource( "/users/home/server/keystore.jks" );
ksp.setPassword( "keystorePassword" );

KeyManagersParameters kmp = new KeyManagersParameters();
kmp.setKeyStore(ksp);
kmp.setKeyPassword( "keyPassword" );

SSLContextServerParameters scsp = new SSLContextServerParameters();
scsp.setClientAuthentication(ClientAuthentication.REQUIRE);
SSLContextParameters scp = new SSLContextParameters();
scp.setServerParameters(scsp);
scp.setKeyManagers(kmp);

SSLContext context = scp.createSSLContext();
SSLEngine engine = scp.createSSLEngine();
```

#### Configuring Different Options on the Client and Server Side

In this example, both the client and server sides share the same custom key store; however, the client side allows any supported cipher suite while the server side will use the default cipher suite filter and exclude any cipher suites that match the patterns *\*NULL.\** and *\*anon.\**.

```
KeyStoreParameters ksp = new KeyStoreParameters();
ksp.setResource( "/users/home/server/keystore.jks" );
ksp.setPassword( "keystorePassword" );

KeyManagersParameters kmp = new KeyManagersParameters();
kmp.setKeyStore(ksp);
kmp.setKeyPassword( "keyPassword" );

FilterParameters filter = new FilterParameters();
filter.getInclude().add( ".*" );

SSLContextClientParameters sccp = new SSLContextClientParameters();
sccp.setCipherSuitesFilter(filter);

SSLContextParameters scp = new SSLContextParameters();
scp.setClientParameters(sccp);
scp.setKeyManagers(kmp);

SSLContext context = scp.createSSLContext();
SSLEngine engine = scp.createSSLEngine();
```

#### Using Camel Property Placeholders

This configuration utility fully supports the use of property placeholders (see [Using PropertyPlaceholder](#)) in all configuration fields. In order to support this feature, the configuration utility objects must be configured with a reference to a Camel context. All of the utility classes except for CipherSuitesParameters and SecureSocketProtocolsParameters provide a setter method for providing the context reference. Do not confuse the lack of a setter on CipherSuitesParameters and SecureSocketProtocolsParameters as an indication that you cannot use property placeholders when configuring these classes. The lack of a setter is an internal implementation detail and full placeholder support is available for both of the configuration classes.

The following example code demonstrates how to create a KeyStore instance based on configuration options provided by the Camel Properties Component and property placeholder support.

```

PropertiesComponent pc = new PropertiesComponent();
pc.setLocation("file:./jsse-test.properties");

CamelContext context = new DefaultCamelContext();
context.addComponent("properties", pc);

KeyStoreParameters ksp = new KeyStoreParameters();
ksp.setContext(camelContext);
ksp.setType("{{keyStoreParameters.type}}");
ksp.setProvider("{{keyStoreParameters.provider}}");
ksp.setResource("{{keyStoreParameters.resource}}");
ksp.setPassword("{{keyStoreParameters.password}}");

KeyStore keyStore = ksp.createKeyStore();

```

## XML Configuration



Note that XML configuration is supported in both Spring and Blueprint format.

### Setting Client Authentication On the Server Side

This configuration sets the server side aspects of the TLS configuration to require client authentication during the handshake process. This configuration uses the default trust store and a custom key store to provide key material for both the server and client sides of the SSLContext.

```

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:camel="http://camel.apache.org/schema/spring"
       xsi:schemaLocation="
         http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
         http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd">

  <camel:sslContextParameters
    id="mySslContext">

    <camel:keyManagers
      keyPassword="keyPassword">
      <camel:keyStore
        resource="/users/home/server/keystore.jks"
        password="keystorePassword"/>
      </camel:keyManagers>

    <camel:serverParameters
      clientAuthentication="WANT"/>

  </camel:sslContextParameters>

</beans>

```

### Configuring Different Options on the Client and Server Side

In this example, both the client and server sides share the same custom key store; however, the client side allows any supported cipher suite while the server side will use the default cipher suite filter and exclude any cipher suites that match the patterns *\*NULL.\** and *\*anon.\**.

```

<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
  xmlns:camel="http://camel.apache.org/schema/blueprint">

  <camel:sslContextParameters
    id="mySslContext">

    <camel:keyManagers
      keyPassword="keyPassword">
      <camel:keyStore
        resource="/users/home/server/keystore.jks"
        password="keystorePassword"/>
      </camel:keyManagers>

    <camel:clientParameters>
      <camel:cipherSuitesFilter>
        <camel:include>.*</camel:include>
      </camel:cipherSuitesFilter>
    </camel:clientParameters>

  </camel:sslContextParameters>

</blueprint>

```

## Using Camel Property Placeholders

This configuration utility fully supports the use of property placeholders (see [Using PropertyPlaceholder](#)) in all configuration fields for XML based configuration as well. In order to support this feature, the configuration utility objects must be configured with a reference to a Camel context. The Spring and Blueprint namespace handlers will automatically inject the reference to the context for you when there is one Camel context in scope. If you have more than one Camel context instance in your XML defined context, you can indicate which context reference to configure by specifying the `camelContextId` attribute in the top-level XML element.

The following example code demonstrates how to create a KeyStore instance based on configuration options provided by the Camel Properties Component and property placeholder support. The Camel context with the ID `example` is used to resolve the property placeholders.

```

<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:camel="http://camel.apache.org/schema/spring"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
    http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd">

  <camel:camelContext id="example"/>

  <camel:camelContext id="example2"/>

  <camel:keyStoreParameters
    id="ksp"
    camelContextId="example"
    resource="{{keyStoreParameters.resource}}"
    type="{{keyStoreParameters.type}}"
    provider="{{keyStoreParameters.provider}}"
    password="{{keyStoreParameters.password}}"/>

</beans>

```