```java
@Bean
public FTPSClient createFTPSClient(FTPSConnectionService
ftpsConnectionService, FTPSConnectionProperties ftpsConnectionProperties) {
    try {
        FTPSClient ftpsClient = new
FTPSClient(ftpsConnectionService.getSSLContext(ftpsConnectionProperties));
        ftpsClient.setConnectTimeout((int)
Duration.ofMinutes(ftpsConnectionProperties.getSessionTimeout()).toMillis());
        ftpsClient.connect(ftpsConnectionProperties.getHost(),
ftpsConnectionProperties.getPort());
        log.info("*********FTPS connection established.************");
        return ftpsClient;
    } catch (IOException ex) {
        log.error("*********FTPS connection failed.************", ex);
    }
    return new
FTPSClient(ftpsConnectionService.getSSLContext(ftpsConnectionProperties));
}
```

```java
@Service
@Data
@Slf4j
@Profile("D")
public class FTPSConnectionService {

    private final SSLConfigurator sslConfigurator;
    private SSLContext sslContext;

    public FTPSConnectionService(SSLConfigurator sslConfigurator) {
        this.sslConfigurator = sslConfigurator;
    }

    public SSLContext getSSLContext(FTPSConnectionProperties
connectionProperties) {
        log.info("Inside the method  :FTPS getSSLContext");
        sslContext =
sslConfigurator.createSSLContext(connectionProperties.getKeystorePath(),
connectionProperties.getTruststorePath(),
connectionProperties.getKeystorePassword(),
connectionProperties.getTruststorePassword(),
connectionProperties.getKeystoreType(),
connectionProperties.getTruststoreType());

        return sslContext;
    }
}
```

```java
@Data
@ConfigurationProperties(prefix = "platform.messaging.ftps")
@NoArgsConstructor
@Profile("D")
public class FTPSConnectionProperties {

    //TLS common properties
    @Value("${platform.messaging.mq.keystore-password}")
    private String keystorePassword;
    @Value("${platform.messaging.mq.truststore-password}")
    private String truststorePassword;
    @Value("${platform.messaging.mq.keystore-path}")
    private String keystorePath;
    @Value("${platform.messaging.mq.truststore-path}")
    private String truststorePath;
    @Value("${platform.messaging.mq.keystore-type:pkcs12}")
    private String keystoreType;
    @Value("${platform.messaging.mq.truststore-type:jks}")
    private String truststoreType;
    @Value("${platform.messaging.mq.autoconfigure}")
    private boolean autoconfigure;

    //FTPS properties
    @Value("${ftps.host}")
    private String host;
    @Value("${ftps.port}")
    private Integer port;
    @Value("${ftps.username}")
    private String username;
    @Value("${ftps.sessionTimeout}")
    private Long sessionTimeout;

}
```

```java
@Slf4j
@Data
public class Bloker {
    private SslParams params;

    public Broker(SslParams params) {
        this.params = params;
    }

    public static void main(String[] args) {
        SpringApplication.run(Bloker.class, args);
    }

    @PostConstruct
    void postConstruct() {
        setTrustStoreParams();
    }

    private void setTrustStoreParams() {
        log.info(String.format("Setting javax properties"));
        System.setProperty("javax.net.ssl.trustStore",
params.trustStorePath);
        System.setProperty("javax.net.ssl.trustStorePassword",
params.trustStorePassword);
    }
}
```

```java
@Component
@ConfigurationProperties("params")
@Data
public class SslParams {
    @Value("${platform.messaging.mq.truststore-path}")
    public String trustStorePath;
    @Value("${platform.messaging.mq.truststore-password}")
    public String trustStorePassword;

    public String getTrustStorePath() {
        return trustStorePath;
    }

    public void setTrustStorePath(String trustStorePath) {
        this.trustStorePath = trustStorePath;
    }

    public String getTrustStorePassword() {
        return trustStorePassword;
    }

    public void setTrustStorePassword(String trustStorePassword) {
        this.trustStorePassword = trustStorePassword;
    }
}
```

```yaml
ftps:
  host: anyhost.host.net
  port: 2121
  username: s2XX -spid
  sessionTimeout: 2
```