**ИНТЕГРАЦИЯ GLASSFISH 4.1 и JTLS 2.0**

Благодарность пользователю *yauhen\_l*

*(*https://github.com/yauhen-l/glassfish-gost-tls*)*

за указания по настройке *glassfish 3* и *JTLS 1.0*

**Задача**: ***создать ГОСТ конфигурацию для порта 8181***

Предполагается, что имеются установленные под учетной записью администратора *glassfish 4.1, JDK 1.8.0\_25, JCP* и *JTLS 2.0*, сняты экспортные ограничения в *JRE* (<https://support.cryptopro.ru/index.php?/Knowledgebase/Article/View/44/6/snjatie-ehksportnykh-ogrnichenijj>), установлен *CSP 3.6 R4* или *4.0* (для проверки подключения через *IE, csptest*).

Далее *<glassfish\_root>* - корневая папка установленного веб-сервера.

Перед выполнением инструкции следует внимательно ознакомиться с руководством по использованию *JCP* и *JTLS*, в частности, с разделами установки, удаления и настройки.

**Важное замечание!** Файл-образец *new\_configuration*/*domain.xml* сконфигурирован для системы Windows.

*Порты: 8080 и 8181 – проверка приложений, 4848 – Административная консоль.*

1. **Подготовка сервера**

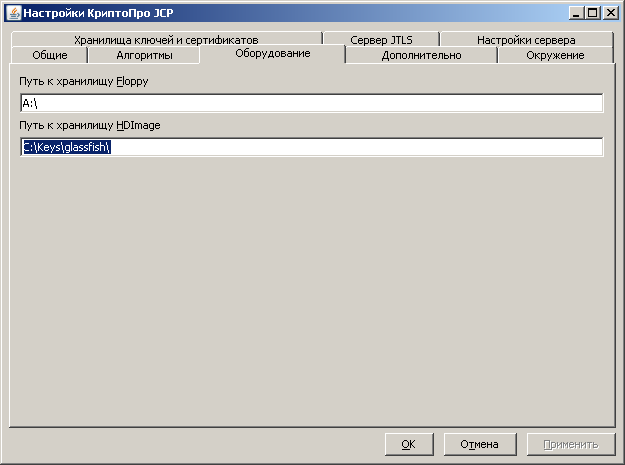
Необходимо:

1. найти файл *<glassfish\_root>\glassfish\domains\domain1\config\logging.properties*, добавить в него строки

*ru.CryptoPro.JCP.tools.JCPLogger.level = INFO  
 ru.CryptoPro.JCP.tools.JCPLogger.handlers = java.util.logging.ConsoleHandler  
  
 ru.CryptoPro.ssl.SSLLogger.level = ALL  
 ru.CryptoPro.ssl.SSLLogger.handlers = java.util.logging.ConsoleHandler*

Для *SSLLogger* можно указать более высокий уровень, например, *INFO* или *FINE*, но *ALL* - максимально детальный. В дальнейшем его можно заменить на *INFO*;

1. запустить панель JCP, открыть вкладку *«Оборудование»*, указать статический путь к папке с контейнерами, чтобы не зависеть от пользователя, например, так:



Поместить туда серверный ключевой контейнер, например, *server* (см. ниже), если он был сгенерирован (получен) в другом месте. Иначе серверный контейнер должен лежать в папке *Crypto Pro* того пользователя, под управлением учетной записи которого запускается сервер;

1. создать серверный ключевой контейнер, например, *server,* с политикой серверной аутентификации в панели JCP и паролем *changeit* на закладке *«Хранилища ключей и сертификатов»*. Сертификат должен иметь политику «аутентификация сервера» и «аутентификация клиента», его CN должно содержать имя сервера или его ip адрес (например, *localhost*). После генерации контейнера нужно сохранить запрос в base64-кодировке и выпустить сертификат в Тестовом УЦ, затем серверный сертификат нужно добавить в контейнер *server*.

**Важно**! Требование создавать контейнер и формировать запрос в панели JCP, а не, например, на странице Тестового УЦ с помощью IE, происходит из того, что код glassfish содержит ошибку: <https://java.net/jira/browse/GLASSFISH-21249>

**UPDATE**: В версии glassfish 4 под названием Payara (<http://www.payara.co/upstream_builds>) ошибка была исправлена, и сертификаты можно создавать как в панели JCP, так и с помощью CSP в IE.

1. создать в панели JCP на закладке *«Хранилища ключей и сертификатов»* хранилище доверенных сертификатов (корневых), например, *trust.store,* с паролем *changeit*, поместить в него корневые сертификаты центров, в которых могут быть выпущены сертификаты клиентов, если на сервере будет использоваться двухсторонняя аутентификация.
2. **Подготовка конфигурации**

В папке *<glassfish\_root>/glassfish/domains/domain1/config* есть файл *domain.xml.* В нем имеются конфигурации *default-config* и *server-config*. Настройка будет производиться для второй. Перед изменением файла *domain.xml* рекомендуется сделать его резервную копию. В качестве примера нужно использовать образец *new\_configuration/domain.xml*.

1. В узле *<config name="server-config">* следует отыскать листенер *http-listener-2* - по умолчанию для него настроен SSL на порту 8181;
2. сформировать примерно такую конфигурацию:

*<protocol name="http-listener-2" security-enabled="true">  
 <http max-connections="250" default-virtual-server="server">  
 <file-cache></file-cache>  
 </http>* ***<ssl ssl3-enabled="false" trust-algorithm="GostX509" classname="ru.cryptopro.jtlsconnector.JTLSProxy" trust-store="path\_to\_trust\_store" cert-nickname="alias" key-store="path\_to\_trust\_store" client-auth-enabled="<client\_need\_auth>"></ssl>*** *</protocol>*

Такая настройка может быть выполнена как вручную, простым редактированием файла, так и в административной консоли *glassfish* в разделе с описанием листенера *http-listener-2* конфигурации *server-config*,на вкладке *SSL*.

Здесь:

*А) ru.cryptopro.jtlsconnector.JTLSProxy* - класс специально собранной SSL-конфигурации для корректной связи с JTLS (будет описан далее);

*Б) path\_to\_trust\_store* - путь к хранилищу доверенных сертификатов, в нашем случае - путь к trust.store, например, «C:\Keys\trust.store»;

*В) client\_need\_auth* - значение true или false, означающее, требует ли сервер клиентский сертификат или нет (односторонняя или двухсторонняя аутентификация).

1. Кроме этого, в разделе *java-config с* перечислением аргументов *JVM,* рядом с листенерами, в список нужно добавить новые аргументы:

*<jvm-options>-Djavax.net.ssl.keyStoreProvider=JCP</jvm-options>  
<jvm-options>-Djavax.net.ssl.keyStoreType=HDImageStore</jvm-options>  
<jvm-options>-Djavax.net.ssl.keyStore=path\_to\_trust\_store</jvm-options>  
<jvm-options>-Djavax.net.ssl.keyStorePassword=changeit</jvm-options>  
<jvm-options>-Djavax.net.ssl.trustStoreProvider=JCP</jvm-options>  
<jvm-options>-Djavax.net.ssl.trustStoreType=HDImageStore</jvm-options>  
<jvm-options>-Djavax.net.ssl.trustStore=path\_to\_trust\_store</jvm-options>  
<jvm-options>-Djavax.net.ssl.trustStorePassword=changeit</jvm-options>  
<jvm-options>-Dcom.sun.security.enableCRLDP=true</jvm-options>*

Аргумент *com.sun.security.enableCRLDP=true* требуется, если на сервере включена двухсторонняя аутентификация (если *client-auth-enabled="true"*).

1. **Создание коннектора**

Для работы *glassfish* и ГОСТ TLS требуется создать специальный коннектор, который будет настраивать ряд параметров модуля JTLS: список сайфер-сюиты, алгоритмы и т.д. Исходные коды проекта коннектора есть в папке *Connector*. Необходимо открыть проект в *NetBeans IDE 8 (JDK 8)*, подключить необходимые зависимости, перечисленные далее, и собрать jar-библиотеку.

*JTLSProxyServerSocketFactory* - это класс фабрики серверных сокетов, расширяющий стандартный класс *GlassfishServerSocketFactory* модуля *security.jar* в *glassfish.* Его функциональность дополнена новыми атрибутами, которые считываются и передаются в класс-предок из специального файла *jtls.properties*, который будет размещен в той же папке проекта (а далее - и модуля *JTLSConnector.jar*), что и сам данный класс *GlassfishServerSocketFactory*.

1. Содержимое *jtls.properties* может быть таким:

*trustStoreProvider=JCP*

*trustStoreType=HDImageStore*

*trustStore=path\_to\_trust\_store*

*trustStorePassword=changeit*

*keyStoreProvider=JCP*

*keyStoreType=HDImageStore*

*keyStore=path\_to\_trust\_store*

*keyStorePassword=changeit*

*keyStoreAlias=*

*protocol=GostTLS*

*protocols=GostTLS*

*algorithm=GostX509*

*clientAuth=true\_or\_false*

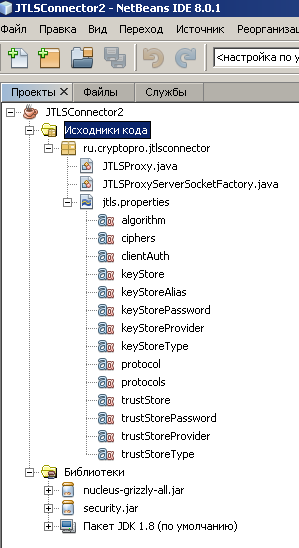
*ciphers=TLS\_CIPHER\_2001*

Параметр *path\_to\_trust\_store* упоминался ранее (путь к хранилищу доверенных сертификатов). Используется только одна сайфер-сюита: *TLS\_CIPHER\_2001*, значениепараметра *keyStoreAlias* указыватьне требуется, значение *ciphers* не используется*.*

При использовании на сервере аутентификации клиента следует установить параметр *clientAuth* равным true.Все значения совпадают с аргументами *JVM* в *domain.xml* в узле конфигурации *server-config;*

1. подключить к проекту зависимости из папки *<glassfish\_root>/glassfish/modules* сервера: библиотеки *nucleus-grizzly-all.jar* и *security.jar.*

Структура проекта будет выглядеть примерно так:



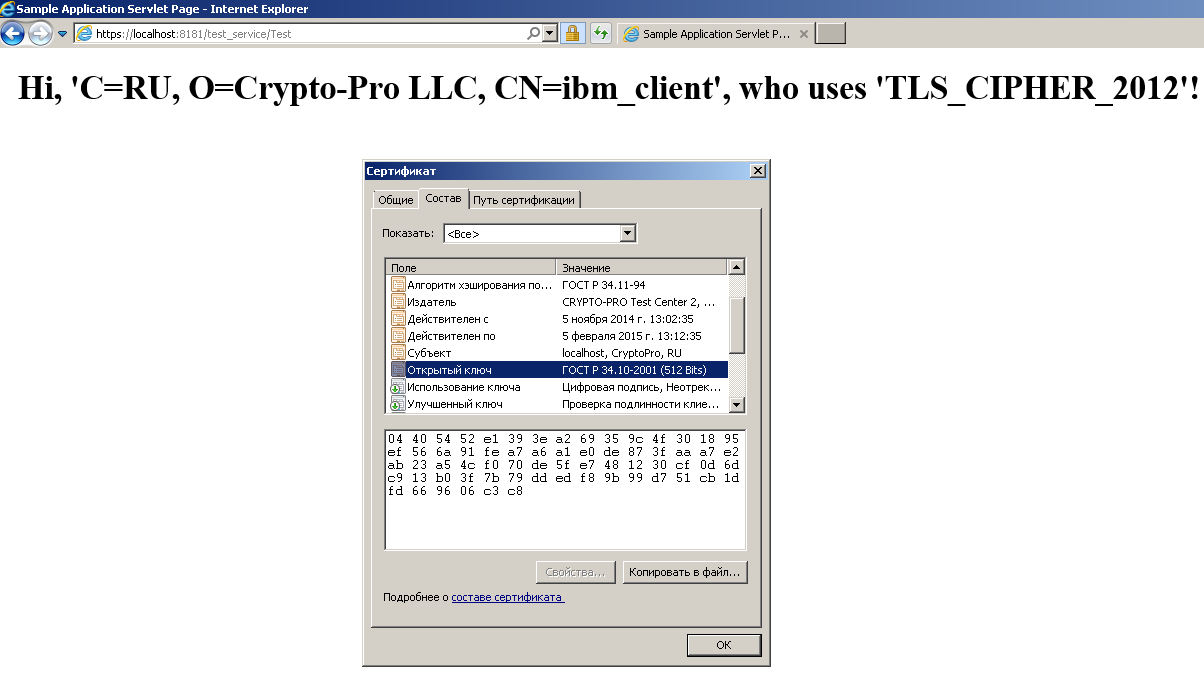
1. произвести сборку *JTLSConnector.jar.* Затем поместить его в *<glassfish\_root>/glassfish/lib;*
2. запустить сервер. После запуска можно проверить файл лога в папке *<glassfish\_root>/glassfish/domains/domain1/logs* на факт передачи параметров командной строки и создания объекта класса *JTLSProxy;*
3. для более детального логирования и диагностики ошибок можно включить уровень логирования FINE у javax.level в файле *<glassfish\_root>/glassfish/domains/domain1/config/logging.properties:*

*javax.level=FINE*

1. **Проверка**

После запуска сервера можно произвести проверку соединений на порту 8181. Для этого нужно:

1. опубликовать тестовое приложение *TestService.war* с контекстом (*root-context*) как */test\_service.* Исходный код и собранное приложение находятся в папке *example/TestService/build*;
2. произвести переход по адресу [*https://localhost:8181/test\_service/Test*](https://localhost:8181/test_service/Test) в браузере IE или выполнить команду в командной строке, используя *csptest*. Например, при включенной клиентской аутентификации в *IE* (*CSP 4.0*) можно увидеть сообщение:

**

При этом использовался установленный ранее клиентский контейнер. При соединении сервером была выбрана максимальная сайфер-сюита *TLS\_CIPHER\_2012* в *JTLS 2.0*, несмотря на явное указание параметра *ciphers* в *jtls.properties*.

**Примечание**: если при включенной аутентификации клиента (*client-auth-enabled)* в *http-listener-2* на сервере подключение не выполняется, одной из вероятных причин может быть установленный в *false* в конфигурационном файле *jtls.properties* (модуль *JTLSConnector*) параметр *clientAuth* – ему следует задать значение *true*.

При односторонней аутентификации (атрибут *client-auth-enabled* получил значение *"false"* в узле *ssl* листенера *http-listener-2* конфигурации *server-config* в файле *domain.xml*) увидим:

