

# Apache 2.4.x

## 2-Way SSL authentication

### Guideline

*Version* 1.0  
*Date:* 02.07.2018  
*Status:* Freigegeben  
*Author:* André Wendel / Andreas Letsche  
*File:* 20180702\_BMW\_Apache24\_Two\_Way\_SSL\_Client\_Auth\_Guideline.docx  
*Pages:* 28

*Copies to:*

#### **History:**

<b>Version</b>	<b>Date</b>	<b>Authors</b>	<b>Changes</b>
0.1	10/9/2014	André Wendel, FG-745	Initial creation
0.2	7/10/2014	André Wendel, FG-745	Adding improvements / comments
0.3	15/10/2014	André Wendel, FG-745	Information about F5 BIG-IP Loadbalancer
0.4	04/05/2015	André Wendel, FG-745	Added information about 2 way SSL architecture
1.0	02/07/2018	Andreas Letsche, FG-831	Minor Corrections

## Contents

<b>2-Way SSL authentication .....</b>	<b>1</b>
<b>1 Architecture.....</b>	<b>4</b>
1.1 System overview .....	4
1.2 One-Way SSL authentication .....	5
1.3 Two-Way SSL authentication .....	6
<b>2 Activate 2 way SSL authentication for client communication .....</b>	<b>7</b>
2.1 Apache configuration.....	7
2.2 Staging configuration.....	9
2.3 Example.....	9
2.3.1 Initial situation .....	9
2.3.2 Scenario.....	10
2.4 Additional information .....	19
<b>3 Activate certificate-based client authentication for Backends .....</b>	<b>20</b>
3.1 Apache HTTP Server .....	20
3.1.1 Configuration.....	20
3.1.2 Staging configuration .....	21
3.2 GlassFish v3.....	22
3.2.1 Configuration.....	22
3.2.2 Staging configuration .....	22
3.3 Example.....	23
<b>4 F5 BIG-IP LoadBalancer.....</b>	<b>24</b>
<b>5 Activate 2 way SSL authentication for backend communication .....</b>	<b>25</b>
5.1 Apache configuration.....	25
5.2 Staging configuration.....	26
<b>Appendix: References and links.....</b>	<b>27</b>
<b>Appendix: Figures.....</b>	<b>28</b>

## Introduction

The document gives an overview of the possibilities to authenticate clients based on a client certificate on the Apache HTTP server via 2-Way SSL authentication or to pass-through the client certificate to the backend (i.e. GlassFish, Payara or Weblogic) application server, to authenticate the client on the backend by the business logic.

In the first part of the document the SSL authentication mechanism for one and two way SSL will be explained and also the certificate structure for two way SSL authentication within a standard web environment will be shown.

The second part explains the technical configuration of the Apache HTTP server to implement two way SSL authentication, between a client and an Apache HTTP server acting as a reverse proxy.

In addition to that technical example in chapter two, chapter three shows how the client certificate can be forwarded to the backend for client certificate based authentication on the backend.

Chapter four of the document describes the necessary changes within the F5 BIG-IP dispatcher, which are required to enable two way SSL authentication on Apache HTTP servers reverse proxies, which are located behind a F5 BIG-IP dispatcher.

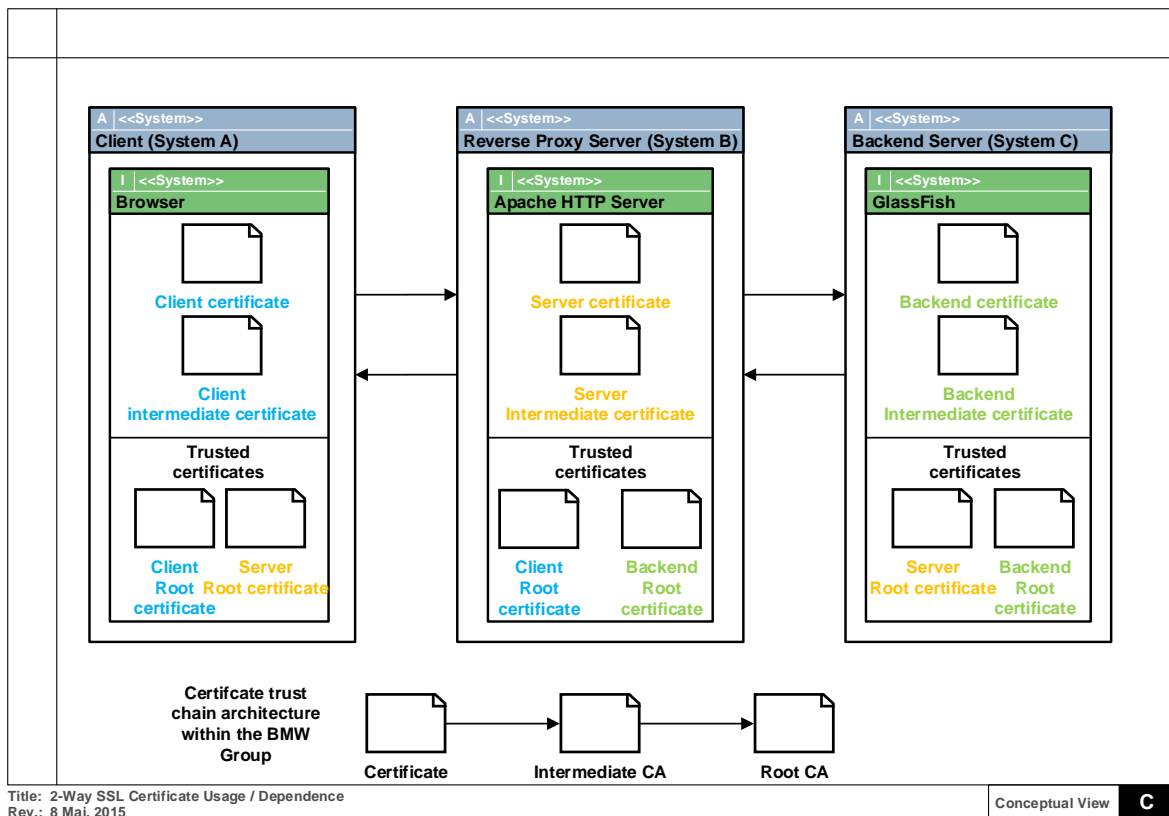
The last chapter of the document describes the technical configuration of the Apache HTTP server reverse proxy, if two way SSL authentication should be enabled between the reverse proxy and the backend system.

# 1 Architecture

The following chapter will give you a short overview how 2-Way SSL authentication will work and explain the certificate architecture in combination of a standard web environment, which consists of a client-, reverse proxy- and a backend-system.

## 1.1 System overview

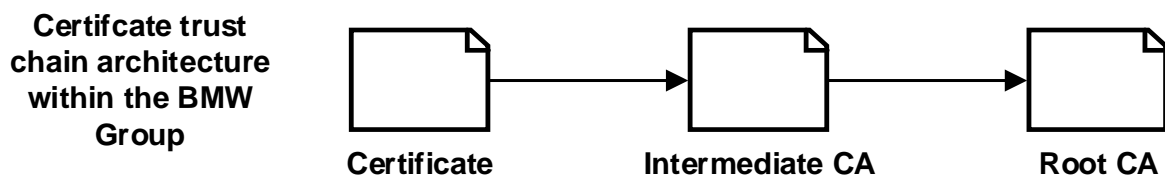
The following architecture (1 Client/Server architecture incl. certificate structure) shows a standard client/server web architecture with an initial certificate structure, which is able to support 2-Way SSL authentication between the client and the reverse proxy server and also between the reverse proxy server and the backend server.



### 1 Client/Server architecture incl. certificate structure

To implement a 2-Way SSL authentication between two systems, both systems need to have SSL communication enabled and provide corresponding certificates for authentication and encryption. In general a certificate of a system is signed and created on a trust chain, this means that certificates are normally not self-signed but rather created and signed from a trusted CA. So every certificates consist of an issuing CA, against that client certificate can be verified.

The certificate trust chain within the BMW Group consists of a Root CA, which signs an intermediate CA on which the client certificate of the system is signed and issued.



### 2 Certificate trust chain

As you can see in figure 1, every systems consists of a certificate and a corresponding intermediate certificate to authenticate and encrypt data against other systems.

Client (System A) certificates are marked **blue**;

Reverse Proxy Server (System B) certificates are marked **yellow**;

Backend Server (System C) certificates are marked **green**;

The systems normally needs to deliver the client and intermediate certificate for the authentication, because the most server systems only hold and trust the certificates of a root CA. These server need both certificates from the client to build and verify the trust chain against a root CA and authenticate the client system. Due that fact every system in figure 1 is equipped with a client and its issuing intermediate certificate.

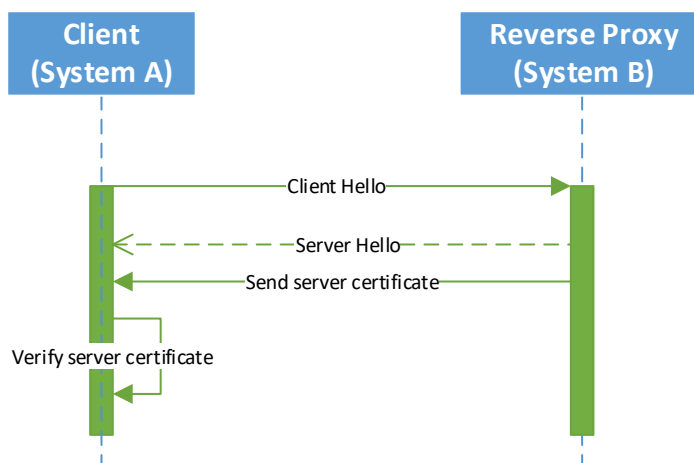
In the most cases within the BMW Group the 2-Way SSL authentication between an external client and the reverse proxy server, located within the DMZ, is needed (communication between system A and system B). For this reason each systems holds and trusts the certificate of the corresponding root CA of the client certificate which should be authenticated. So in our case the client trusts the root CA certificate of reverse proxy and trusts the root CA of the client system certificate (see corresponding/equal colors figure 1).

The following chapters first introduce the standard one-way SSL authentication and based on that the two-way SSL authentication will be explained.

## 1.2 One-Way SSL authentication

The following example shows how one-way SSL authentication works between a client and a server, in our case between the client and the reverse proxy, but the same example could also be done between the reverse proxy and the backend.

One-way SSL authentication enables the application operating as the SSL client to verify the identity of the application operating as the SSL server. One way SSL authentication is done by the most clients automatically (e.g. browsers) and the required server certificates are installed in every web- and application server within the BMW Group, which are provided by the central IT.



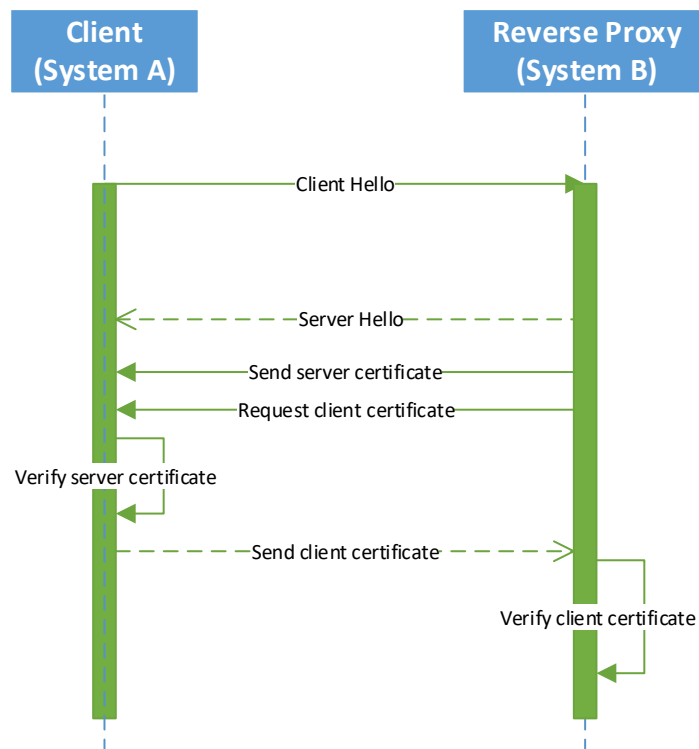
### 3 One way SSL authentication example

During the one way SSL authentication the client systems first sends a “Client Hello” to the server system, and the server system answers the “Client Hello” with a “Server Hello”. After that “Server Hello” the server send automatically his server and intermediate certificate to the client, so that the client is able to authenticate and encrypt the future data within the server communication. After the client receives the client and intermediate certificate of the server he is able to verify the trust chain and accept or deny the server based on the certificate information.

### 1.3 Two-Way SSL authentication

The following example shows how two-way SSL authentication works between a client and a server, in our case between the client and the reverse proxy, but the same example could also be done between the reverse proxy and the backend.

In two-way SSL authentication, the SSL client application verifies the identity of the SSL server application, and then the SSL server application verifies the identity of the SSL-client application.



#### 4 Two way SSL authentication example

The two SSL authentication is based on the one way SSL authentication, but in addition to the one way SSL authentication the server systems demands the client certificate chain (client and intermediate certificate) from the client, after sending his own certificate for verification to the client.

## 2 Activate 2 way SSL authentication for client communication

This chapter describes the 2-Way SSL authentication mechanism/configuration of the Apache HTTP server and shows an example configuration for the Apache HTTP server V2.4.

All shown examples and configurations work also for the Apache HTTP server V2.2, but with the difference that the configuration must be performed directly in the httpd.conf file and does not take place on the staging concept of the Apache HTTP server V2.4.

**Note:** Within the standard BMW Group web architecture the web server cluster is normally located behind a F5 BIG-IP load balancer. If your web server is located behind an F5 BIG-IP load balancer, some additional changes must be performed within the F5 BIG-IP configuration. For details about that changes, please have a detailed look into chapter 3 "[F5 BIG-IP LoadBalancer](#)".

### 2.1 Apache configuration

Necessary directives to activate 2-Way SSL authentication within the Apache HTTP server 2.4 are the following ones:

#### SSLVerifyClient:

This directive sets the Certificate verification level for the Client Authentication. In this case from the four possible configurations only the value "require" is interesting, because with "require" the client has to present a valid certificate.

Example:

```
SSLVerifyClient require
```

See also, [http://httpd.apache.org/docs/2.4/mod/mod\\_ssl.html#sslverifyclient](http://httpd.apache.org/docs/2.4/mod/mod_ssl.html#sslverifyclient).

#### SSLVerifyDepth:

The depth actually is the maximum number of intermediate certificate issuers, i.e. the number of CA certificates which are max allowed to be followed while verifying the client certificate. A depth of 0 means that self-signed client certificates are accepted only, the default depth of 1 means the client certificate can be self-signed or has to be signed by a CA which is directly known to the server.

Based on the BMW standard pki services the certificate trust chain normally consist of a ROOT-CA certificate, an INTERMEDIATE-CA certificate and the CLIENT certificate. All client certificates are normally signed by the corresponding INTERMEDIATE-CA certificate, which itself is signed by the corresponding ROOT-CA certificate. So the trust chain at BMW uses two "CA"-certificates to verify the client certificate, so in the standard case a verify depth of "2" should be enough to verify BMW internal client certificates.

Example:

```
SSLVerifyDepth 2
```

See also, [http://httpd.apache.org/docs/2.4/mod/mod\\_ssl.html#sslverifydepth](http://httpd.apache.org/docs/2.4/mod/mod_ssl.html#sslverifydepth).

### **SSLCACertificateFile:**

This directive sets the all-in-one file where you can assemble the certificates of Certification Authorities (CA) whose clients you deal with. These are used for Client Authentication; at least the ROOT-CA certificate should be part of the certificate file.

Example:

```
SSLCACertificateFile ${APACHE_PROJ}/project-data/certs/ca.pem
```

See also, [http://httpd.apache.org/docs/2.4/mod/mod\\_ssl.html#sslcertificatefile](http://httpd.apache.org/docs/2.4/mod/mod_ssl.html#sslcertificatefile).

### **Require (previously SSLRequire):**

This directive tests whether an authenticated user is authorized according to a particular authorization provider and the specified restrictions i.e. this directive can be used to check certificate properties against specified values, so that not all client certificates can access the server or a defined location.

Example:

```
Require expr %{SSL_CLIENT_S_DN_O} == "BMW" and %{SSL_CLIENT_S_DN_OU} in  
{"FG-745"} and %{SSL_CLIENT_S_DN_CN} in {"clientCert"}
```

See also, [http://httpd.apache.org/docs/current/mod/mod\\_authz\\_core.html#require](http://httpd.apache.org/docs/current/mod/mod_authz_core.html#require).



## 2.2 Staging configuration

The following project allows are necessary to configure a server wide client certificate authentication (no directory based client certificate authentication configuration).

```
<projectallows>
  <allow>serverconfig.sslconfig</allow>
  <allow>serverconfig.sslconfig.sslverifyclient</allow>
  <allow>serverconfig.sslconfig.sslverifydepth</allow>
  <allow>serverconfig.sslconfig.sslcacertificatefile</allow>
</projectallows>
```

The corresponding values can be configured inside the 10\_ProjectConfig.xml file of the project as follows

```
<serverconfig>
  <sslconfig>
    <sslverifyclient>require</sslverifyclient>
    <sslverifydepth>2</sslverifydepth>
    <sslcertificatefile>${APACHE_PROJ}/project-
data/certs/ca.pem</sslcertificatefile>
  </sslconfig>
</serverconfig>
```

Within the BMW configuration only the ROOT-CA should be configured within the directive "SSLCACertificateFile" (all BMW ROOT CA's are normally located under /www/certs/) and the client certificates should be delivered with the corresponding INTERMEDIATE-CA to complete the chain for the verification between client and Apache HTTP server 2.4.

Normally the client certificate and the INTERMEDIATE-CA get delivered as a PKCS12-file, which includes the client certificate, the client certificate key and the INTERMEDIATE-CA. These file can be imported into browsers, on OS level or into the Java KeyStore, so that browsers or applications can use these certificates to authenticate and communicate with the Apache HTTP Server 2.4.

The client sends with each request the client certificate and INTERMEDIATE-CA to the Apache HTTP server, so that the server is able to verify the client certificate together with the INTERMEDIATE-CA based on the ROOT-CA of the server and authenticate the request.

BMW client certificates including the corresponding INTERMEDIATE-CA can be ordered through the BMW PKI service <http://pki.muc/>.

## 2.3 Example

### 2.3.1 Initial situation

Apache HTTP server configured with 2-Way SSL authentication based on a client certificate authentication.

The intermediate-signed client certificates and CA's to build up a trusted certificate chain, based on the following certificates:

File-Name	Certificate-Type	Signed-By	CommonName
ca.cert.pem	Root-CA	-	testCA
intermediate.cert.pem	Intermediate-CA	Root-CA	interCA
www.example.com.cert.pem	Client-Certificate	Intermediate-CA	clientCert

Based on the certificates above a PKCS12 file "www.example.com-chain.pkcs12" was created, which consists of the following certificates and keys and is used for the import of the client certificate and intermediate CA into the browser.

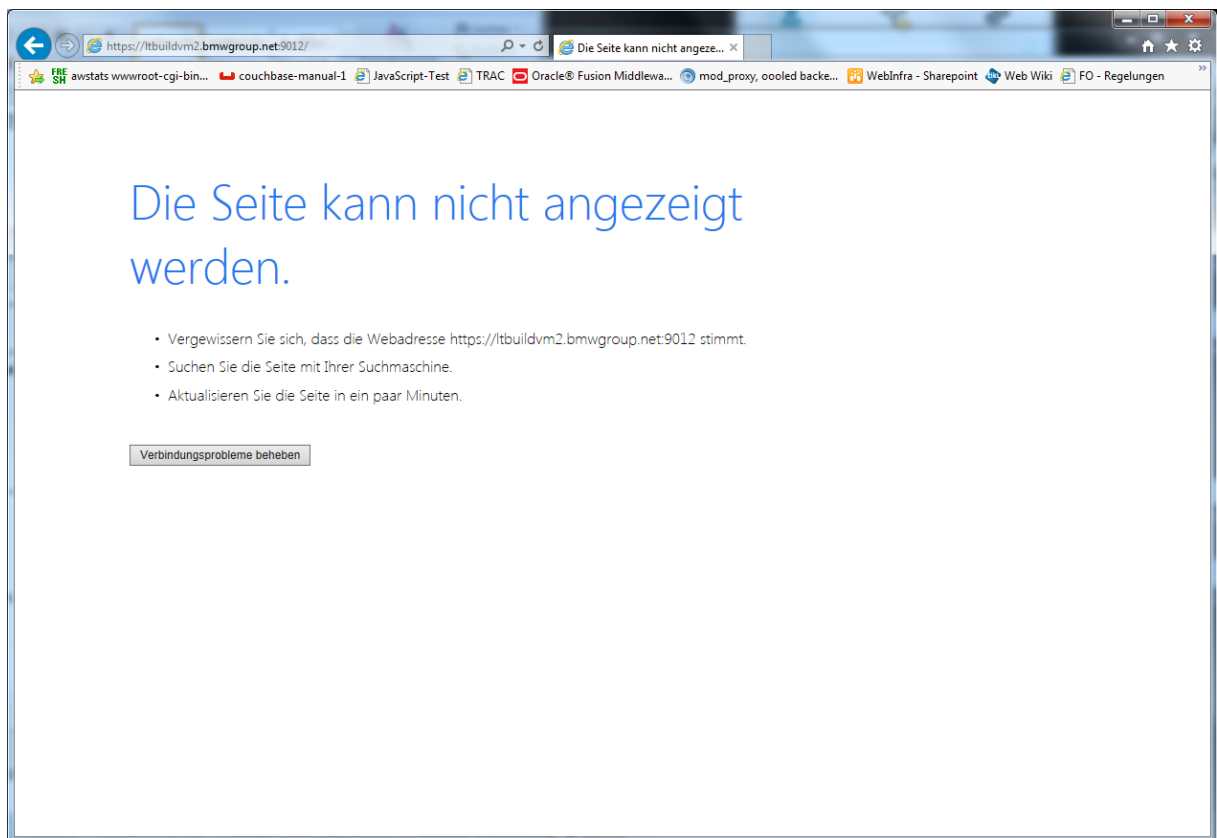
File	Type
www.example.com.cert.pem	Client-Certificate
www.example.com.key.pem	Client-Certificate-Key
intermediate.cert.pem	Intermediate-CA

Based on the certificates and files previously described the Apache HTTP server runs with the following server wide configuration.

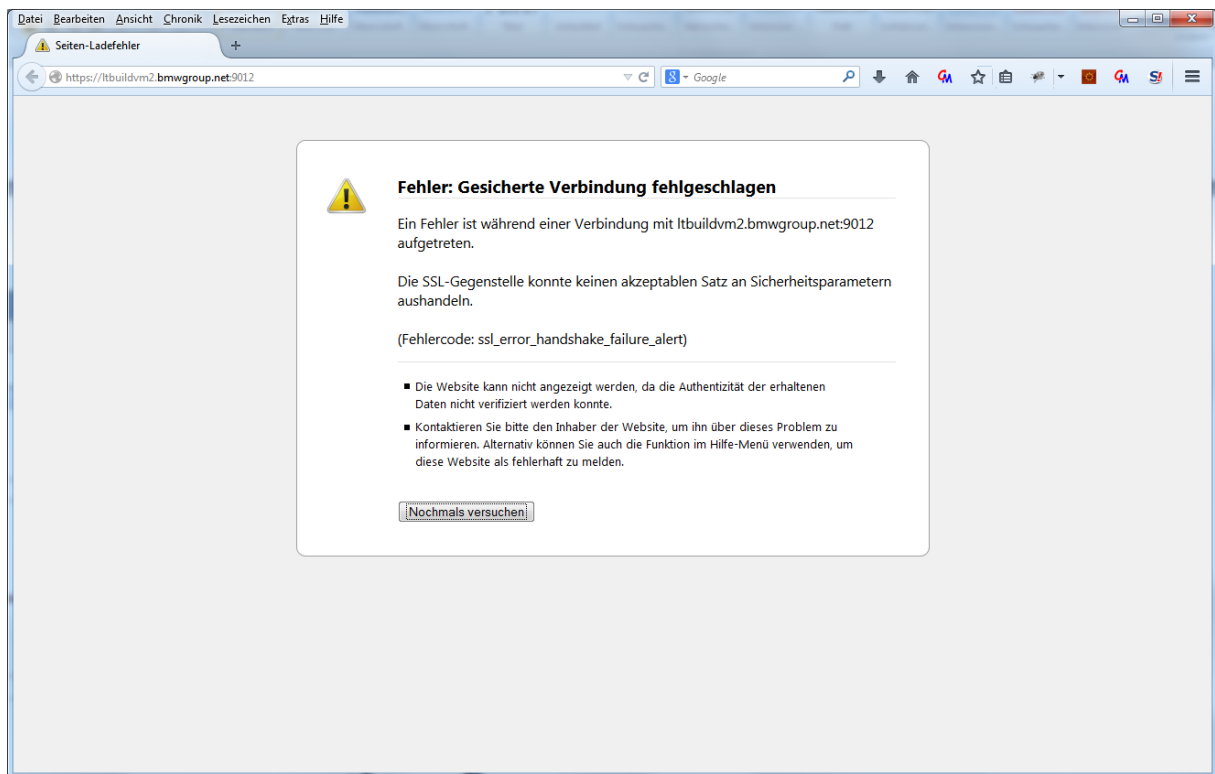
```
SSLVerifyClient require
SSLVerifyDepth 2
SSLCACertificateFile ${APACHE_PROJ}/project-data/certs/ca.cert.pem
```

### 2.3.2 Scenario

With the current configuration the client is not able to connect to the Apache HTTP server. If the user tries to access the URL of the server, the browser will display one of the following pages.



5 Internet Explorer

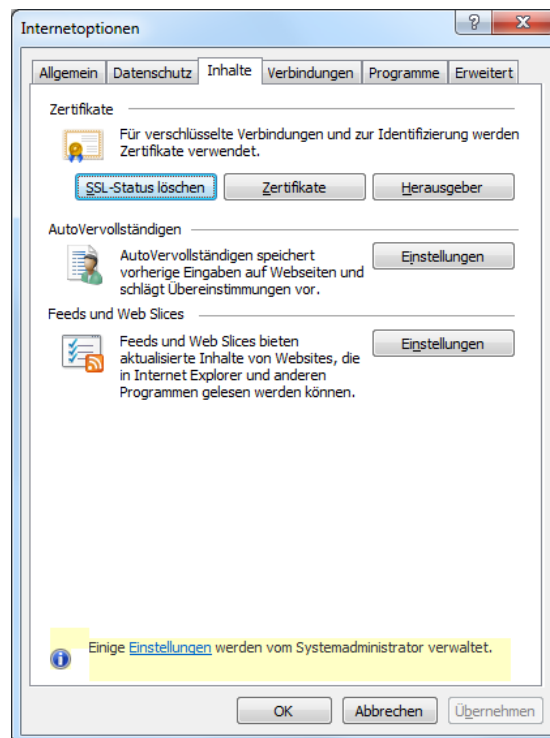


#### 6 Mozilla Firefox

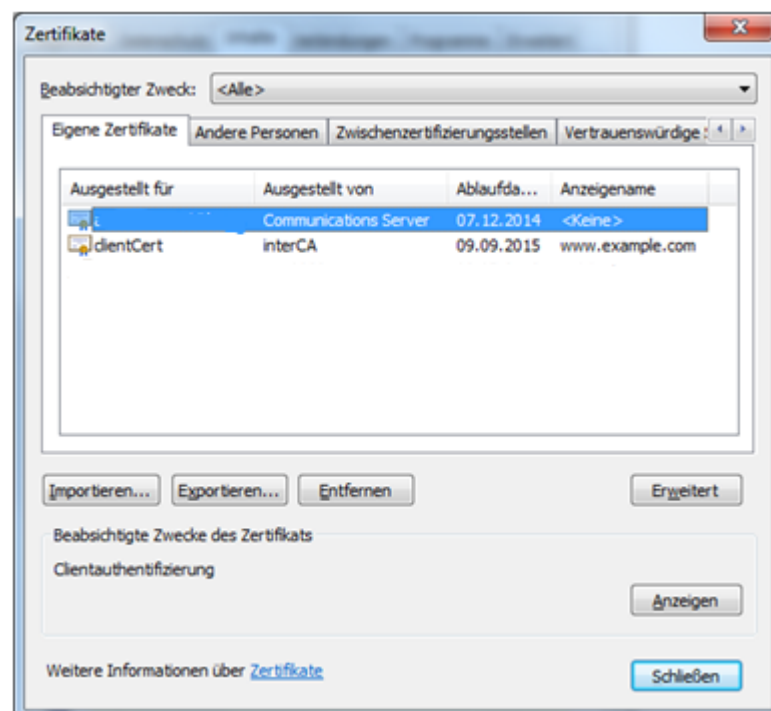
This screen displays the user that the browser is not able to establish a secured connection, because of a missing certificate handshake. This handshake is not possible, because the certificate store of the client doesn't contain the corresponding client certificates, which compare to the root-ca of the Apache HTTP server.

In this case we can import the necessary certificates into the key store of the corresponding browser so that the browser is able to provide the necessary client certificates. For this reason we have to import the PKCS12 file into certificate stores of the Internet Explorer or Mozilla Firefox browser, which can be done via the certificate manager. Because of the reason that the Apache HTTP server only trusts the Root-CA, we need to include the client certificate and the corresponding Intermediate-CA.

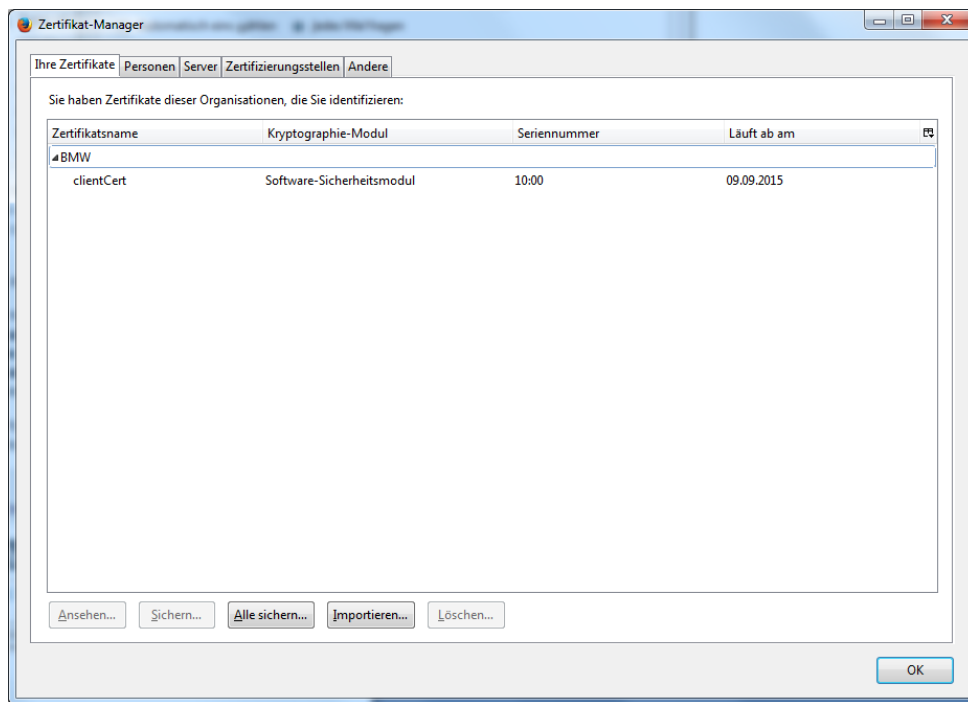
After the import of the PKCS12 file the certificate manager displays your client certificate, within the tab "Inhalte → Zertifikate → Eigene Zertifikate" of the Internet Explorer or for Mozilla Firefox within the tab "Ihre Zertifikate".



7 Internet Explorer: Certificate store

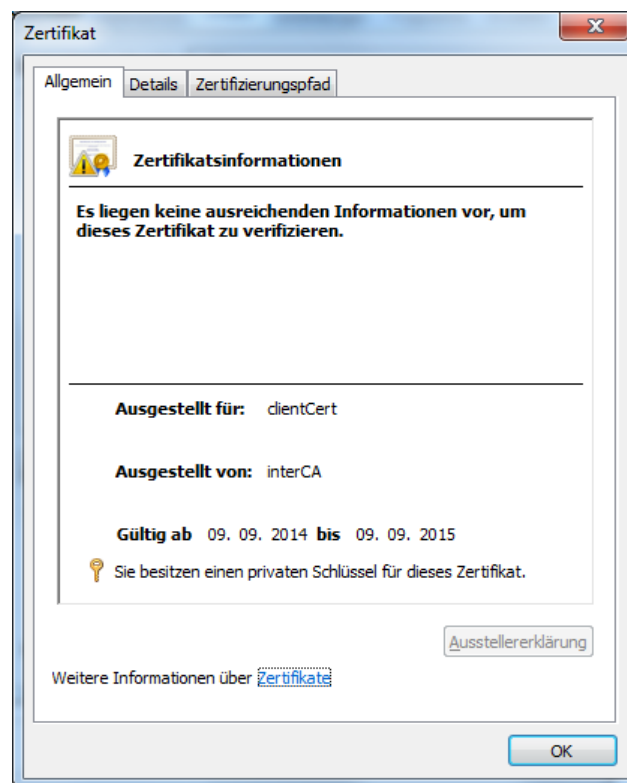


8 Internet Explorer: Tab "Eigene Zertifikate"

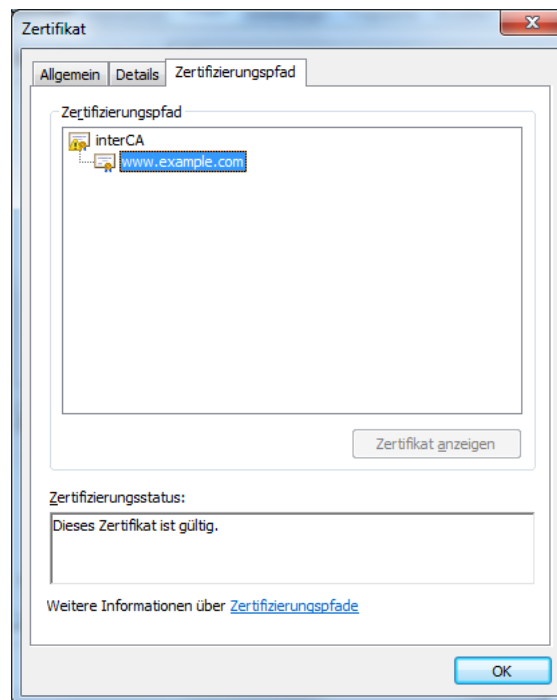


9 Mozilla Firefox: Tab "Zertifikate"

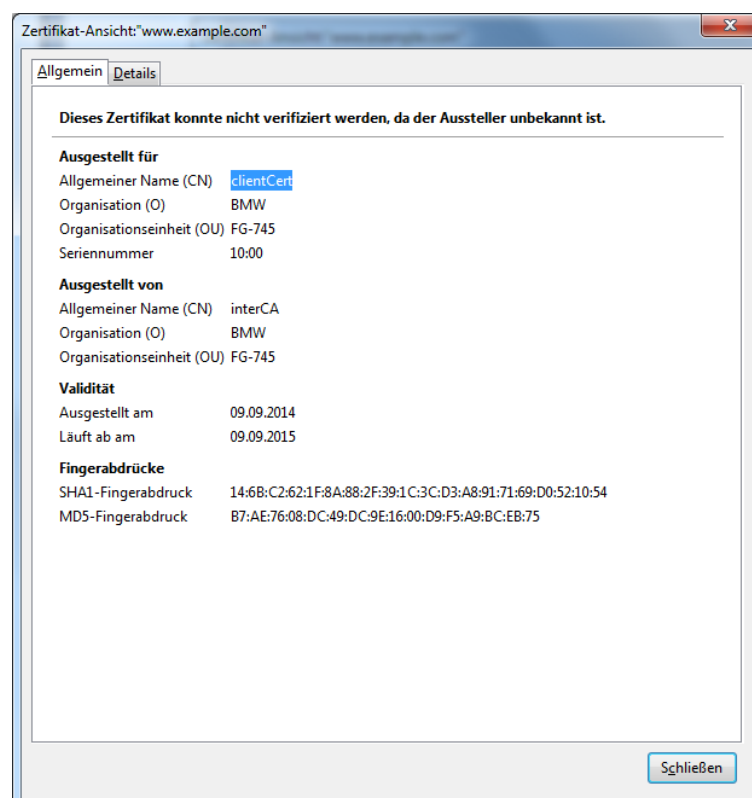
By opening the details of the client certificate, we can see that the client certificate with the common name "clientCert", which is signed by "interCA" was imported.



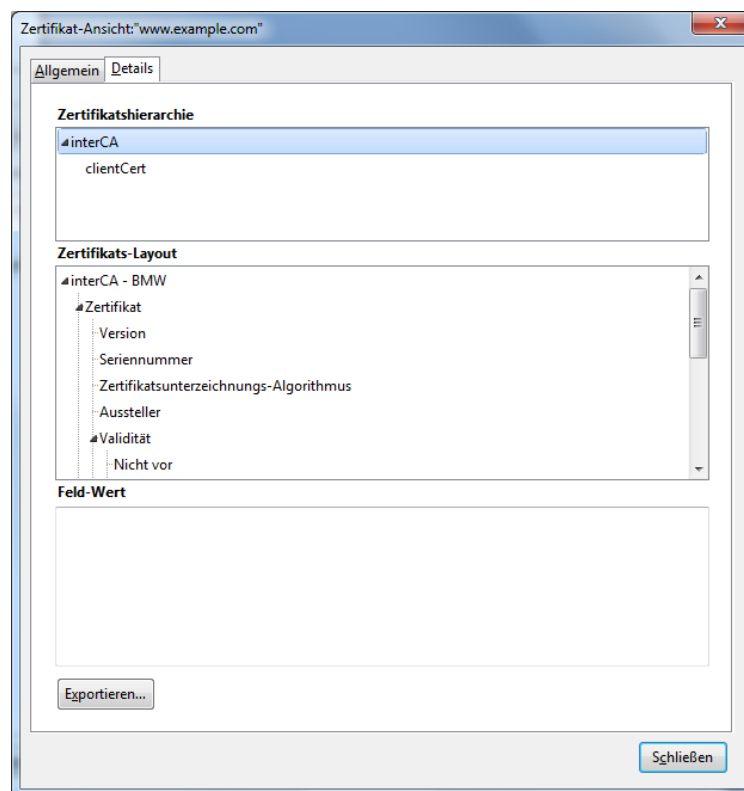
10 Internet Explorer: General view client certificate



11 Internet Explorer: Certificate chain client certificate

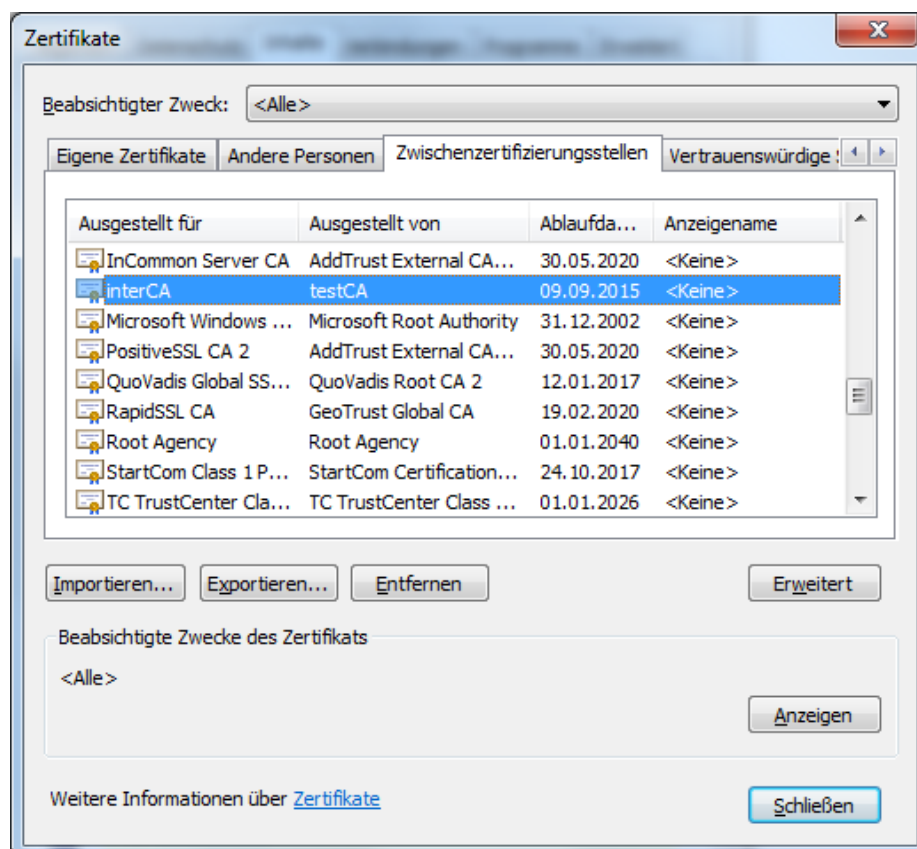


12 Mozilla Firefox: General view client certificate

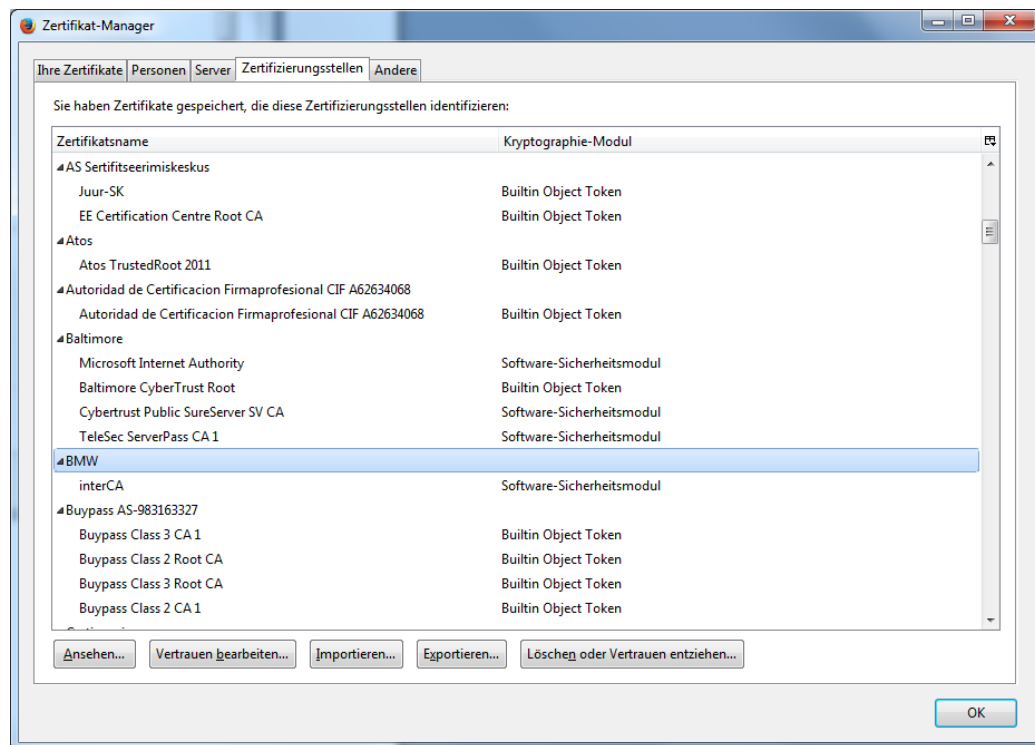


13 Mozilla Firefox: Detailed view client certificate

If we now have also a look into the tab “Zwischenzertifizierungsstellen” for the Internet Explorer or into “Zertifizierungsstellen” for Mozilla Firefox, we will also find the added Intermediate-CA, which was also part of the imported PKCS12 file.

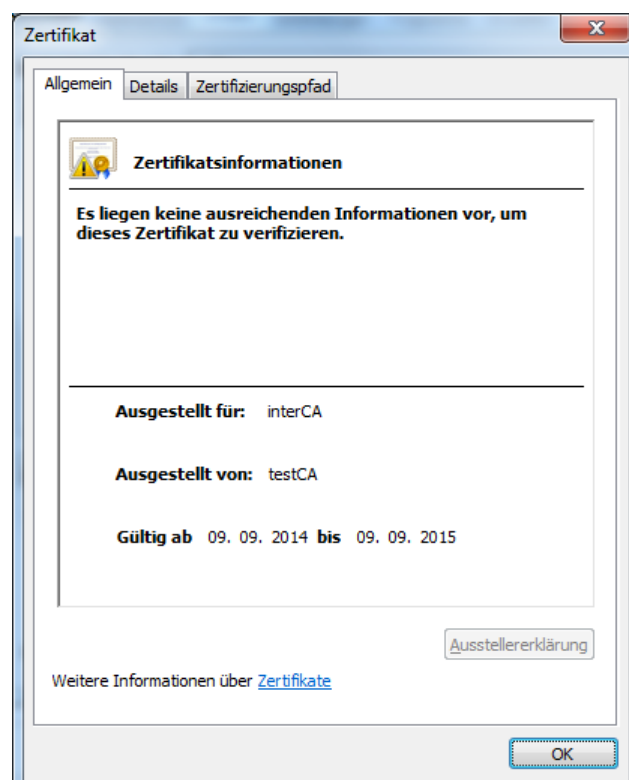


14 Internet Explorer: Intermediate certificate



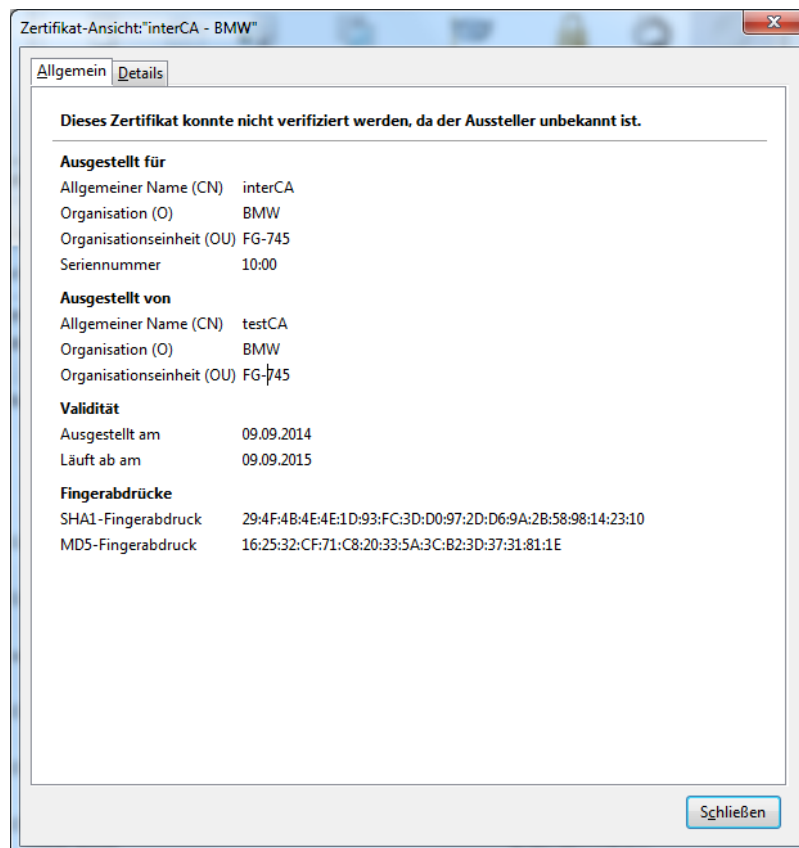
15 Mozilla Firefox: Intermediate certificate

Within the details of the Intermediate-CA “interCA”, we see the correct common name and also the information that the certificate was signed by the RootCA “testCA”, which is configured as SSLCACertificateFile within the Apache HTTP Server.



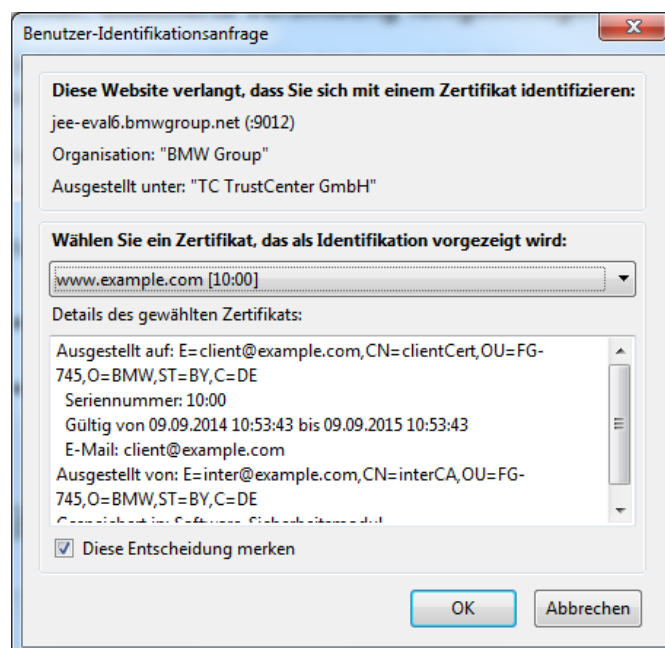
16 Internet Explorer: General view intermediate certificate





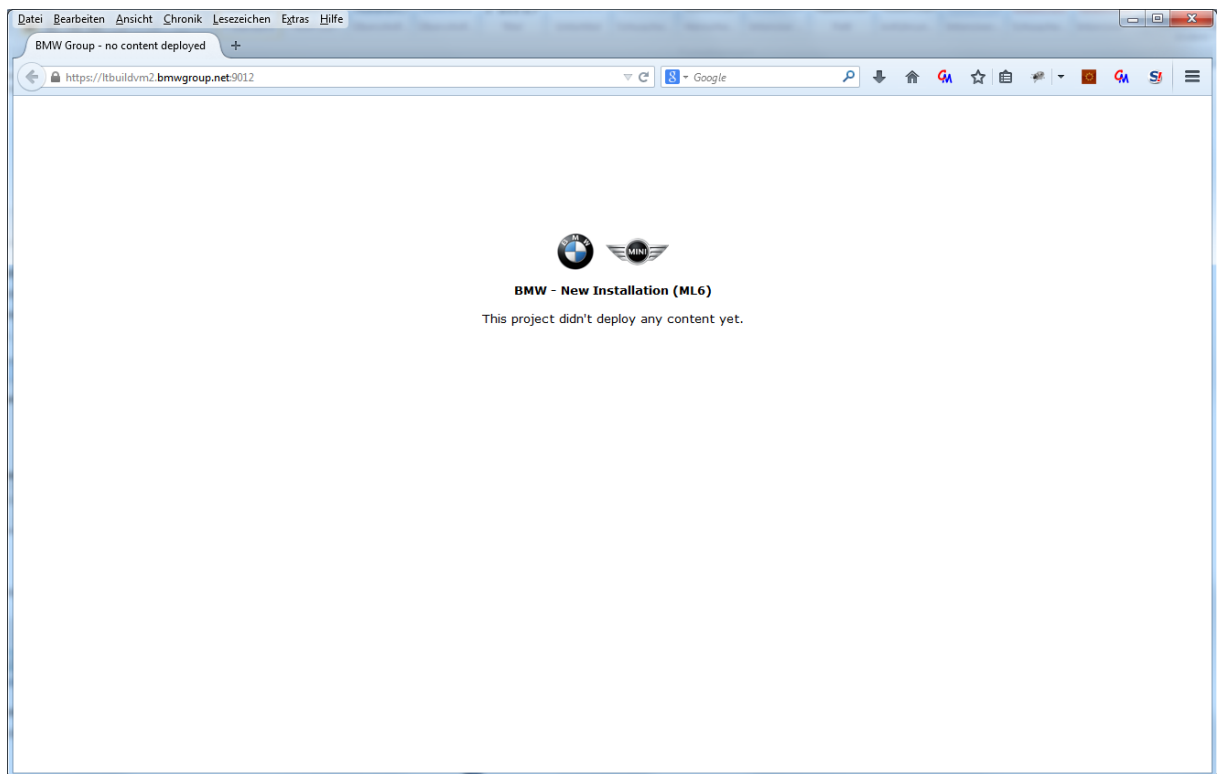
17 Mozilla Firefox: General view intermediate certificate

After the import of the necessary certificates the client is able to connect to the Apache HTTP server. If the client now makes a new request to connect to the Apache HTTP server a new popup will be displayed by the Firefox browser (for the Internet Explorer NO popup will be displayed, because the certificate selection will be done automatically if a corresponding certificate is available), which enables the user to select a client certificate for the usage of authentication at the Apache HTTP server.



18 Mozilla Firefox: Client certificate dialog

If the user now selects the correct client certificate, in this example the certificate with the common name “clientCert”, the browser is able to connect to the URL and displays the content correctly.



19 Successfull client certificate authentication

## 2.4 Additional information

The previously in chapter 1.1 Apache Configuration introduced directive “Require” can be used to implement an additional SSL certificate check based on parameters of the client certificate. For example, with this directive, it is possible to check if the client certificate CN is issued for a special client or organization.

Based on the previous scenario a possible check could be if the organization of the client certificate is the BMW Group and the organizational unit is FG-745, and only if the client certificate matches these parameters, it will get access to the backend.

Based on this information a possible configuration could be the following

```
Require expr %{SSL_CLIENT_S_DN_O} == "BMW" and %{SSL_CLIENT_S_DN_OU} in {"FG-745"} and %{SSL_CLIENT_S_DN_CN} in {"clientCert"}
```

Due to the fact that the “Require”-directive is only available on directory-level of the Apache HTTP server, we need to include the configuration inside a “Location” or “Directory”-directive. Due to the fact that we want to secure the whole web server in this example, we choose the directory path “/”, which leads to the following configuration.

```
<Location / >  
Require expr %{SSL_CLIENT_S_DN_O} == "BMW" and %{SSL_CLIENT_S_DN_OU} in {"FG-745"} and %{SSL_CLIENT_S_DN_CN} in {"clientCert"}  
</Location>
```

The corresponding values can be configured inside the 10\_ProjectConfig.xml file of the project as follows

```
<serverconfig>  
  <sectionsconfig>  
    <location uri="/">  
      <authconfig>  
        <requireconfig>  
          <require entityname="expr"  
            value="%{SSL_CLIENT_S_DN_O} eq &quot;BMW&quot; and  
              %{SSL_CLIENT_S_DN_OU} in {&quot;FG-745&quot;} and  
              %{SSL_CLIENT_S_DN_CN} in  
                {&quot;clientCert&quot;}"/>  
        </requireconfig>  
      </authconfig>  
    </location>  
  </sectionsconfig>  
</serverconfig>
```

For this configuration the following projectallows within the 02\_ApacheConfigOperationsExtra.xml are required:

```
<projectallows>  
  <allow>serverconfig.sectionsconfig</allow>  
  <allow>serverconfig.sectionsconfig.location</allow>  
  <allow>serverconfig.sectionsconfig.location.authconfig</allow>  
  <allow>serverconfig.sectionsconfig.location.authconfig.requireconfig</allow>  
  <allow>serverconfig.sectionsconfig.location.authconfig.requireconfig.require</allow>  
</projectallows>
```

## 3 Activate certificate-based client authentication for Backends

Certificate-based client authentication gives the backend system the ability to authenticate clients based on their certificate, so no additional login with username and password is necessary to authenticate a client on the backend.

### 3.1 Apache HTTP Server

To enable the backend to authenticate clients by the certificate-based client authentication method, the necessary information (client certificate) must be forwarded by the Apache HTTP server to the backend.

To enable the Apache HTTP server to forward the client certificate and some additional information to the backend the following settings must be implemented.

#### 3.1.1 Configuration

To use certificate-based client authentication on the backend system the Apache HTTP server in front of the backend must be configured to support 2-Way SSL client authentication, like it is described within the chapter "2 Activate".

In addition to the enabling of the 2-Way SSL client authentication the Apache HTTP server must forward the following header variables to the GlassFish backend:

- Proxy-keysize (Key size of SSL cipher)
- Proxy-ip (IP address of the client)
- Proxy-auth-cert (X.509 certificate)

For Apache HTTP Server v2.4.x this can be done by the usage of the following settings:

```
SSLOptions +StdEnvVars +ExportCertData
RequestHeader append Proxy-keysize %{SSL_CIPHER_USEKEYSIZE}s
SetEnvIf Remote_Addr "(.*)" my_ip=$1
RequestHeader append Proxy-ip %{my_ip}e
RequestHeader append Proxy-auth-cert %{SSL_CLIENT_CERT}s
RequestHeader edit Proxy-auth-cert "-----BEGIN\ CERTIFICATE----- (.*) ---
---END\ CERTIFICATE-----" $1
RequestHeader edit* Proxy-auth-cert " " ""
```

Additional information about the used modules and directives are available at:

**SetEnvIf:**

[http://httpd.apache.org/docs/2.4/mod/mod\\_setenvif.html#setenvif](http://httpd.apache.org/docs/2.4/mod/mod_setenvif.html#setenvif)

**SSLOptions:**

[http://httpd.apache.org/docs/current/mod/mod\\_ssl.html#ssloptions](http://httpd.apache.org/docs/current/mod/mod_ssl.html#ssloptions)

**RequestHeader:**

[http://httpd.apache.org/docs/current/mod/mod\\_headers.html#requestheader](http://httpd.apache.org/docs/current/mod/mod_headers.html#requestheader)

### 3.1.2 Staging configuration

The following project allows are necessary to configure the pass-through of the client certificate to the backend:

```
<projectallows>
    <allow>serverconfig</allow>
    <allow>serverconfig.sslconfig</allow>
    <allow>serverconfig.sslconfig.ssloptions</allow>
    <allow>serverconfig.sslconfig.ssloptions.ssloption</allow>
    <allow>serverconfig.headerconfig</allow>
    <allow>serverconfig.headerconfig.requestheaderblock</allow>
    <allow>serverconfig.headerconfig.requestheaderblock.requestheader</allow>
    <allow>serverconfig.setenvconfig</allow>
    <allow>serverconfig.setenvconfig.setenvifblock</allow>
    <allow>serverconfig.setenvconfig.setenvifblock.setenvif</allow>
</projectallows>
```

The corresponding values for the settings can be configured inside the 10\_ProjectConfig.xml file of the project as follows:

```
<!-- SSL certificate configuration -->
<sslconfig>
    <ssloptions>
        <ssloption>+stdenvvars</ssloption>
        <ssloption>+exportcertdata</ssloption>
    </ssloptions>
</sslconfig>

<!-- Adding certificate info into request header -->
<headerconfig>
    <requestheaderblock>
        <requestheader headername="Proxy-keysize" method="append"
value="%{SSL_CIPHER_USEKEYSIZE}s"/>
        <requestheader headername="Proxy-ip" method="append"
value="%{my_ip}e"/>
        <requestheader headername="Proxy-auth-cert" method="append"
value="%{SSL_CLIENT_CERT}s"/>
        <requestheader headername="Proxy-auth-cert" method="edit"
value="-----BEGIN\ CERTIFICATE----- (.*) -----END\ CERTIFICATE-----"
replacement="$1"/>
        <requestheader headername="Proxy-auth-cert" method="edit*"
value=""; ";" replacement=""";"/>
    </requestheaderblock>
</headerconfig>
```

```
<!-- Set IP address as environment variable -->
<setenvconfig>
    <setenvifblock>
        <setenvif attribute="Remote_Addr" regex=""(.*)""
envvariable="my_ip=$1"/>
    </setenvifblock>
</setenvconfig>
```

## 3.2 GlassFish v3

### 3.2.1 Configuration

To enable the client-certificate based authentication within the GlassFish application server additional parameters need to be set, so that the application server is able to handle the additional header information passed by the Apache HTTP server and also the application within the GlassFish must be configured to support client-certificate based authentication.

To enable the application to support client-based authentication the setting “auth-method” within the web.xml must be bind to the value “CLIENT-CERT”.

As example the following must be configured:

```
<login-config>
    <auth-method>CLIENT-CERT</auth-method>
</login-config>
```

If the application is ready to support client-certificate based authentication, the GlassFish application server needs to be configured in addition to the application. It is necessary to set the following properties within the GlassFish application server cluster with the corresponding values:

- **auth-pass-through-enabled** = true
- **proxyHandler** = com.sun.enterprise.web.ProxyHandlerImpl

For additional information about the two properties, see also “Oracle GlassFish Server 3.1-3.1.1 High Availability Administration Guide”.

### 3.2.2 Staging configuration

The following permission of dotted names are necessary to configure the client-certificate based authentication on the backend:

- cluster-config.network-config.protocols.protocol.http-listener-2.http.auth-pass-through-enabled
- cluster-config.http-service.property.proxyHandler

The corresponding values for the dotted names can be configured inside the 10\_ProjectConfig.xml file of the GlassFish application server as follows:

```
<dotted-name name="cluster-config.network-config.protocols.protocol.http-
listener-2.http.auth-pass-through-enabled" value="true" />

<dotted-name name="cluster-config.http-service.property.proxyHandler"
value="com.sun.enterprise.web.ProxyHandlerImpl" />
```

### 3.3 Example

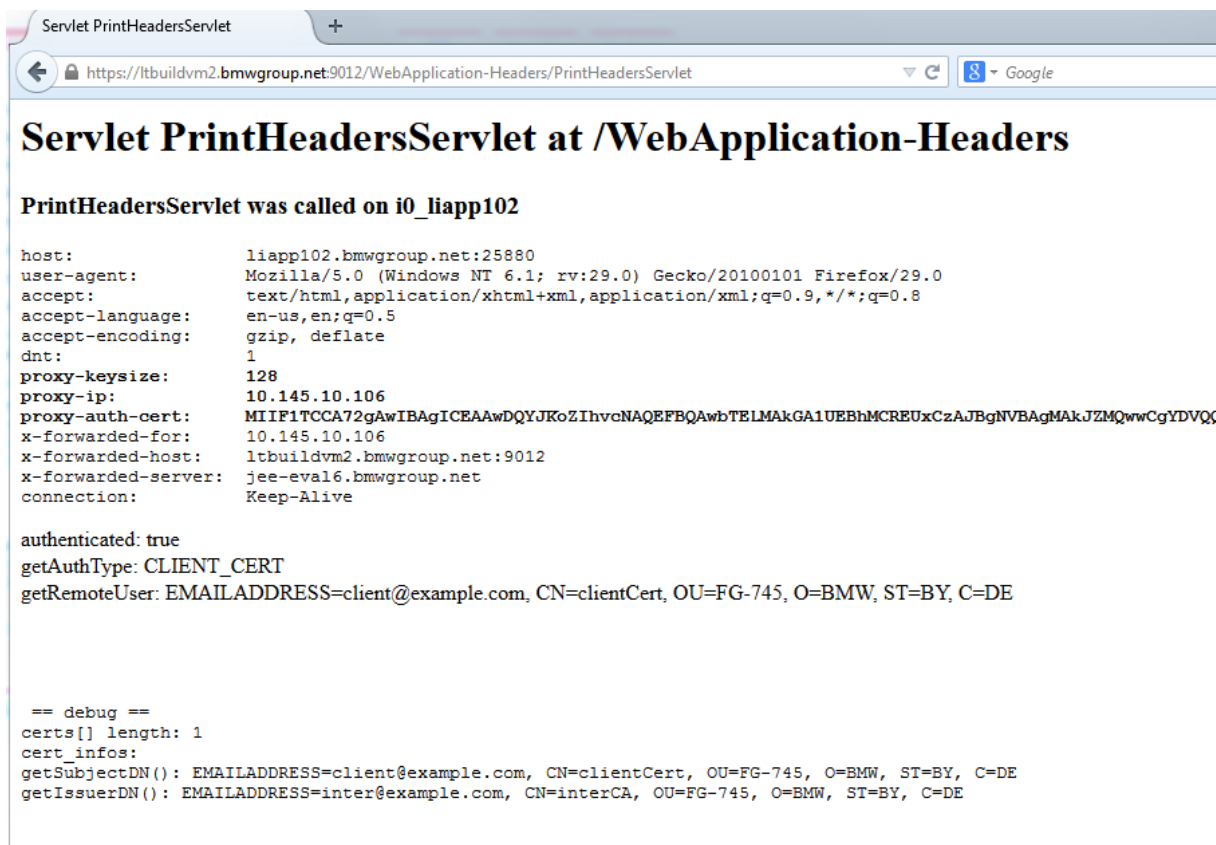
If the Apache HTTP server, the GlassFish application server and the application is configured correctly to enable client-certificate based authentication, the client should be authenticated by the certificate and the servlet context variables within the application should be set as expected:

- request.getAuthType() returns the string identifier for client certificate authentication "CLIENT\_CERT"
- request.getRemoteUser() returns subjectDN of the client certificate

The following example is setup as follows:

- Apache HTTP Server: ltbuildvm2.bmwgroup.net
- GlassFish Application server: liapp102.bmwgroup.net
- Java web application: WebApplication-Headers

By accessing the web application "WebApplication-Headers", which is a small application to print all headers of the request and some additional information of the servlet context, on the GlassFish application server via the proxy server, all information is processed correctly and the client is authenticated within the servlet context.



Servlet PrintHeadersServlet

https://ltbuildvm2.bmwgroup.net:9012/WebApplication-Headers/PrintHeadersServlet

## Servlet PrintHeadersServlet at /WebApplication-Headers

**PrintHeadersServlet was called on i0\_liapp102**

```
host:                liapp102.bmwgroup.net:25880
user-agent:          Mozilla/5.0 (Windows NT 6.1; rv:29.0) Gecko/20100101 Firefox/29.0
accept:              text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
accept-language:     en-us,en;q=0.5
accept-encoding:     gzip, deflate
dnt:                 1
proxy-keysize:       128
proxy-ip:            10.145.10.106
proxy-auth-cert:     MIIF1TCCA72gAwIBAgICEAAwDQYJKoZIhvcNAQEFBQAwbTElMAkGA1UEBhMCREUxCzAJBgNVBAGMAkZJMzQwCgYDVQQL
x-forwarded-for:     10.145.10.106
x-forwarded-host:    ltbuildvm2.bmwgroup.net:9012
x-forwarded-server:  jee-eval6.bmwgroup.net
connection:          Keep-Alive

authenticated: true
getAuthType: CLIENT_CERT
getRemoteUser: EMAILADDRESS=client@example.com, CN=clientCert, OU=FG-745, O=BMW, ST=BY, C=DE

== debug ==
certs[] length: 1
cert_infos:
getSubjectDN(): EMAILADDRESS=client@example.com, CN=clientCert, OU=FG-745, O=BMW, ST=BY, C=DE
getIssuerDN(): EMAILADDRESS=inter@example.com, CN=interCA, OU=FG-745, O=BMW, ST=BY, C=DE
```

20 Example application for backend authentication

## 4 F5 BIG-IP LoadBalancer

Within the standard provisioning process of the Java ML's and the ReverseProxy ML v1.0 the initial load balancing in front of the Apache HTTP Server cluster is done by an F5 BIG-IP load balancer. During the provisioning process of this load balancer an HTTPS\_MONITOR is set up, to check if the configured Apache HTTP server back ends are available and working.

With the activation of certificate based client side authentication this monitor is no longer able to establish a working HTTP connection and for that reason no requests will be forwarded to one of the back ends. For the F5 BIG-IP the back ends are not available anymore and will be deactivated, because the required client certificates to authenticate the requests of the F5 BIG-IP are not available within the monitor.

This problem characterizes that the Apache HTTP server responds by a direct access of the web server URL, but the response breaks if the access is routed over a F5 BIG-IP load balancer.

In this case the HTTPS\_MONITOR has to be replaced by a TCP\_MONITOR for the effected URL on the F5 BIG-IP load balancer. To change the monitor, please open a change task for web operations within the ITSM suite, with the order to replace the „**itpsweb\_monitor\_https**“-monitor by a „**itpsweb\_monitor\_tcp**“-monitor for the effected F5 BIG-IP URL.



## 5 Activate 2 way SSL authentication for backend communication

This chapter describes the 2-Way SSL authentication mechanism/configuration of the Apache HTTP server, when the server acts as a reverse proxy client and shows an example configuration for the Apache HTTP server V2.4. In this case the reverse proxy server must hand over his certificate to the backend server, so that the backend is able to verify the client certificate of the Apache HTTP server.

### 5.1 Apache configuration

Necessary directives to activate client certificate file to enable 2-Way SSL authentication for the Apache HTTP server 2.4 client are the following ones:

#### **SSLProxyMachineCertificateChainFile:**

This directive sets the all-in-one file where you keep the certificate chain for all of the client certs in use. This directive will be needed if the remote server presents a list of CA certificates that are not direct signers of one of the configured client certificates.

This referenced file is simply the concatenation of the various PEM-encoded certificate files. Upon startup, each client certificate configured will be examined and a chain of trust will be constructed. Normally this file contains the corresponding intermediate certificates of the server certificate e.g. for internal servers the issuing BMW intermediate certificate or for internet webserver the issuing GlobalSign certificate.

**Security warning:** If this directive is enabled, all of the certificates in the file will be trusted as if they were also in SSLProxyCACertificateFile.

Example:

```
SSLProxyMachineCertificateChainFile ${APACHE_PROJ}/project-  
data/certs/proxyCA.pem
```

See also, [http://httpd.apache.org/docs/2.4/mod/mod\\_ssl.html#sslproxymachinecertificatechainfile](http://httpd.apache.org/docs/2.4/mod/mod_ssl.html#sslproxymachinecertificatechainfile).

#### **SSLProxyMachineCertificateFile:**

This directive sets the all-in-one file where you keep the certificates and keys used for authentication of the proxy server to remote servers.

This referenced file is simply the concatenation of the various PEM-encoded certificate files, in order of preference. Within the BMW Group architecture this file contains the server certificate and key of the current webserver room, see also SSLCertificateFile and SSLCertificateKeyFile.

Use this directive alternatively or additionally to SSLProxyMachineCertificatePath.

**Security warning:** Currently there is no support for encrypted private keys

Example:

```
SSLProxyMachineCertificateFile ${APACHE_PROJ}/project-data/certs/proxy.pem
```

See also, [http://httpd.apache.org/docs/2.4/mod/mod\\_ssl.html#sslproxymachinecertificatefile](http://httpd.apache.org/docs/2.4/mod/mod_ssl.html#sslproxymachinecertificatefile).

#### **SSLProxyMachineCertificatePath:**

This directive sets the directory where you keep the certificates and keys used for authentication of the proxy server to remote servers.

The files in this directory must be PEM-encoded and are accessed through hash filenames.

**Security warning:** Currently there is no support for encrypted private keys

Example:

```
SSLProxyMachineCertificatePath ${APACHE_PROJ}/project-data/certs/
```

See also, [http://httpd.apache.org/docs/2.4/mod/mod\\_ssl.html#sslproxymachinecertificatepath](http://httpd.apache.org/docs/2.4/mod/mod_ssl.html#sslproxymachinecertificatepath).

## 5.2 Staging configuration

The following project allows are necessary to configure the server to provide certificates for the client certificate authentication with another server.

```
<projectallows>
  <allow>serverconfig.sslproxymachinecertificatechainfile</allow>
  <allow>serverconfig.sslproxymachinecertificatefile</allow>
  <allow>serverconfig.sslproxymachinecertificatepath</allow>
</projectallows>
```

The corresponding values can be configured inside the 10\_ProjectConfig.xml file of the project as follows

```
<serverconfig>
  <sslproxymachinecertificatechainfile>
    ${APACHE_PROJ}/project-data/certificate/proxyCA.pem
  </sslproxymachinecertificatechainfile>
  <sslproxymachinecertificatefile>
    ${APACHE_PROJ}/project-data/certificate/proxy.pem
  </sslproxymachinecertificatefile>
</serverconfig>
```

BMW client certificates including the corresponding INTERMEDIATE-CA can be ordered through the BMW PKI service <http://pki.muc/>.

## Appendix: References and links

Official example for 2-Way SSL client authentication:  
[http://httpd.apache.org/docs/2.4/ssl/ssl\\_howto.html](http://httpd.apache.org/docs/2.4/ssl/ssl_howto.html)

External example:  
<http://linuxconfig.org/apache-web-server-ssl-authentication>

General overview about certificate handling/creation:  
<https://jamielinux.com/blog/category/CA/>

BMW PKI service:  
<http://pki.muc>

BMW Web operations:  
<http://web.bmwgroup.net>

Information from "stackoverflow.com" for Payara 4  
<https://stackoverflow.com/questions/35849851/glassfish-4-certificate-based-client-authentication>

## Appendix: Figures

1 Client/Server architecture incl. certificate structure	4
2 Certificate trust chain	4
3 One way SSL authentication example	5
4 Two way SSL authentication example	6
5 Internet Explorer	10
2 Mozilla Firefox	11
3 Internet Explorer: Certificate store	12
4 Internet Explorer: Tab "Eigene Zertifikate"	12
5 Mozilla Firefox: Tab "Zertifikate"	13
6 Internet Explorer: General view client certificate	13
7 Internet Explorer: Certificate chain client certificate	14
8 Mozilla Firefox: General view client certificate	14
9 Mozilla Firefox: Detailed view client certificate	15
10 Internet Explorer: Intermediate certificate	15
11 Mozilla Firefox: Intermediate certificate	16
12 Internet Explorer: General view intermediate certificate	16
13 Mozilla Firefox: General view intermediate certificate	17
14 Mozilla Firefox: Client certificate dialog	17
15 Successfull client certificate authentication	18
16 Example application for backend authentication	23