How-to-Guide:
Apache as Reverse Proxy for Fiori Applications
Document History:

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Authored By</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Kiran Kola</td>
<td>Architect Engineer</td>
</tr>
</tbody>
</table>
# Table of Contents

1. Business Scenario ......................................................... 4  
2. Prerequisites ................................................................. 4  
3. Fiori Application Architecture ........................................... 5  
4. Apache HTTP Server Installation ........................................ 6  
5. Communication Protocol Scenarios ...................................... 9  
6. Monitoring settings for Apache Server .................................. 25  
7. Troubleshooting communication issues .................................. 27
1. Business Scenario

SAP supports following proxy solutions for Fiori applications:

- Apache Server as Reverse proxy
- SAP Web Dispatcher

Apache Server:

In this guide, we will illustrate how to set up an Apache server as reverse proxy for Fiori applications. Plain http, one-way https and two-way https communication scenarios are covered.

2. Prerequisites

All the server names used in this documentation are used to demonstrate end-to-end technical scenarios and for mockup purposes only. Following are the prerequisites and software details:

ECC/Gateway Server:

- Used ECC 6.0 EhP7 SP 7 with HANA DB (Gateway Embedded Approach)-
  Host Name: mo-06a18f128.mo.sap.corp
- For Fiori Installation and Configuration, refer http://scn.sap.com/docs/DOC-41598
- For ABAP SSL, refer http://scn.sap.com/docs/DOC-53536

Apache server

A typical usage of reverse proxy is to provide mobile user access to SAP Gateway servers that are behind the corporate firewall so Apache HTTP server is installed in a DMZ area.

- Apache Version: Version 2.4
- Apache Server Node: ushplvm1383.phl.sap.corp
- Notepad++  http://notepad-plus-plus.org/

Assumptions:

- For SSL configuration, self-signed certificates are not used in below examples; we used internal SAP CA for signing all the servers and client certificates
- Fiori application with SSL setup (https) is already deployed prior to this setup. However, all the major high level steps are covered in this documentation
- For this implementation, we considered embedded approach (where backend & gateway components on the same system). Same steps applied for central hub approach (where gateway is separated from backend)
- Same configuration steps can be applied for Fiori Client based applications
3 Fiori Application Architecture

Below diagram is the sample Fiori application architecture for ECC/Gateway using Apache as the reverse proxy solution.

Fiori Application Architecture using Apache as Reverse Proxy

In the following sections, we will provide configuration steps to setup plain HTTP, one-way HTTPs and mutual HTTPS authentication using X.509 certificates.
4. Apache HTTP Server Installation

In this section, Apache server installation and configuration is illustrated in the following steps:

1. Download Apache
2. Configure Apache Server

1. Use the link to download the Apache HTTP Server: [http://www.apachelounge.com/download/](http://www.apachelounge.com/download/)

Version used: httpd-2.4.9-win64-VC11

Prerequisite:
Download and install the Windows C++ 2012 runtime from Microsoft.com

We installed Apache in C:\Apache24, so extracted the ZIP file to the root of the C: drive. Apache can be installed anywhere on your system, but you will need to change the configuration file paths accordingly.

Within the folder, you will see following folder structure:
Configure Apache:

a) `cd apache24\bin`

Note: `httpd.exe -k install -n "Apache2.4"` (this installs apache as a service)

Port Conflict scenario: Because Apache cannot share the same port with another TCP/IP application, you may need to stop, uninstall or reconfigure certain other services before running Apache (for example IIS). In default, server listens on port 80 and you can change the port in `httpd.conf` file.

b) Edit `httpd.conf` file using Notepad++, located under `<Drive>\Apache24\conf`.

c) To activate, uncomment following modules in `httpd.conf` file:

Typical proxy server will need to enable several modules. Those relevant for proxying and load balancing are as follows:

- LoadModule proxy_module modules/mod_proxy.so
  - The core module deals with proxy infrastructure and configuration and managing a proxy request.
- LoadModule proxy_http_module modules/mod_proxy_http.so
  - This module handles fetching documents with HTTP and HTTPS.
- LoadModule proxy_connect_module modules/mod_proxy_connect.so
  - This handles the CONNECT method for secure (SSL) tunneling.
- LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
  - `mod_proxy_balancer` implements clustering and load-balancing over multiple backends.
- LoadModule slotmem_shm_module modules/mod_slotmem_shm.so
  - Memory provider which provides for creation and access to a shared memory segment.
• LoadModule proxy_html_module modules/mod_proxy_html.so
  o This rewrites HTML links into a proxy’s address space.
• LoadModule headers_module modules/mod_headers.so
  o This modifies HTTP requests and response headers.
• LoadModule lbmethod_byrequests_module modules/mod_lbmethod_byrequests.so
  o Distribute the requests among the various workers
• LoadModule ssl_module modules/mod_ssl.so
  o This module provides SSL v2/v3 and TLS v1 support for the Apache HTTP Server

Apache server setup is completed. Next sections focus on various communication protocols supported by Apache server for accessing Fiori applications.
Communication

5. Communication Protocol Scenarios

In this section, following protocol communication scenarios for Apache Server are covered:

1. HTTP
2. one-way HTTPS
3. two-way HTTPS (so-called Mutual authentication or X.509 client certificate authentication)

Scenario 1: In this section, Apache as reverse proxy using HTTP communication is covered:

1. Configure httpd.conf for plain HTTP communication
2. Restart Apache Server
3. Verify http communication
4. Testing Fiori URL using Apache with HTTP (unsecured communication – not recommended for productive usage)

Proxy can be easily achieved by simply writing the below two rules in your httpd.conf file.

- **Proxypass**: This directive asks the apache server to fetch data from Gateway Node
- **ProxyPassReverse**: This directive rewrites the original URL when the traffic is send back.

1. In the following HTTP examples, Apache server, usphlvm1383.phl.sap.corp is mapped to following SAP Gateway Node on port 8800:

- mo-06a18f128.mo.sap.corp:8800

Listen 80
<VirtualHost *:80>
ProxyPreserveHost On
AllowEncodedSlashes On
ServerName usphlvm1383.phl.sap.corp
ProxyPass / http://mo-06a18f128.mo.sap.corp:8800/ nocanon
ProxyPassReverse / http://mo-06a18f128.mo.sap.corp:8800/
ErrorLog "C:/Apache24/logs/error.log"
</VirtualHost>

2. Restart Apache Server

3. Verify http communication

Validate the configuration by opening a browser and testing these URLs:

Enter SAP backend gateway credentials.

URL should return a page with this information:

![Server reached successfully](image)

4. Testing Fiori URL using Apache with HTTP. Port 80 is the default http port.


![Fiori URL using Apache Server URL (Secured)](image)

**Scenario 2**: In this section, Apache as reverse proxy using one-way HTTPS communication is covered:

1. SAP Gateway SSL Preparation
2. SSL preparation for Apache server
3. Install Trusted Certificates
4. Configure `httpd.config` for one-way HTTPS communication
5. Restart Apache Server
6. Verify communication
7. Testing Fiori URL using Apache Server URL (Secured)

Reverse proxy, and SAP Gateway Server each use their own certificate; you can create or sign these certificates from one root certificate. In one-way SSL scenario, only the client authenticates the server. This means that the public cert of the Apache server needs to be configured in the trust store of the SAP Gateway Server.
1. SAP Gateway SSL Preparation

Refer following link for SSL setup on ABAP based systems:

http://scn.sap.com/docs/DOC-53536

2. SSL Preparation for Apache Server

The OpenSSL is used to generate an RSA Private Key and CSR (Certificate Signing Request). It can also be used to generate self-signed certificates which can be used for testing purposes or internal usage.

Depending on your operating system, download the OpenSSL software from following link:

https://www.openssl.org/related/binaries.html

a) Generate RSA

`openssl genrsa -des3 -out server.key 2048`

Enter pass phrase twice to generate server.key: s3pAdmin

b) Create CSR file

1. Set the environment variable: `set OPENSSL_CONF=c:\OpenSSL-Win64\bin\openssl.cfg`
2. Issue this command:

   `openssl req -sha256 -out ApacheServer.csr -new -newkey rsa:2048 -nodes -keyout server.key`

Country Name: CA
State or Province Name: ONTARIO
Locality Name: TORONTO
Organization Name: SAP
Organizational Unit Name: COE
Common Name: USPHLM1383.PHL.SAP.CORP
Email Address: Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password:
An optional company name:

c) Generate signed Certificate

For production environments, the Certificate Signing Request that you generated can be submitted to a CA to create a certificate signed by the CA.

d) Remove Passphrase from Key

Apache will ask for the pass-phrase each time the web server is started. Obviously this is not necessarily convenient so you can remove pass phrase from the generated key by following commend:

1. `copy server.key server.key.org`
2. `openssl rsa -in server.key.org -out server.key`

Result is new RSA server.key is generated.
e) Copy `server.key` and `ApacheServer.crt` to Apache conf directory. The location of this directory will differ depending on where Apache is installed.

3. Installing Trusted Certificates

**SAP Gateway**

Using STRUST, upload CA root certificate and Apache Server host as the trusted certificates (required for mutual authentication)

**Apache Platform**

Install CA root certificate onto the Apache server

**For example:**

Right click on the certificate and add it to trusted Root Certificate as shown below.

4. Configuring SSL properties in `httpd.conf`

In the following example, `https://usphlvm1383.phl.sap.corp:443/` is mapped to following SAP Gateway Node:

- `mo-06a18f128.mo.sap.corp:8400`

Listen 443

```
<VirtualHost *:443>
  SSLEngine On
  SSLProxyEngine On
  ProxyRequests Off
  ProxyPreserveHost On
  SSLProxyCheckPeerCN off
```
SSLProxyCheckPeerName off
AllowEncodedSlashes On
SSLCertificateFile /Apache24/conf/ApacheServer.crt
SSLCertificateKeyFile /Apache24/conf/server.key
SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown
ServerName usphlvm1383.phl.sap.corp
ProxyPass / http://mo-06a18f128.mo.sap.corp:8400/ nocanon
ProxyPassReverse / http://mo-06a18f128.mo.sap.corp:8400/
ErrorLog "C:/Apache24/logs/error.log"
TransferLog "C:/Apache24/logs/access.log"
</VirtualHost>

5. Restart Apache server.

6. Verify one-way HTTPS Scenario:

Validate the configuration by opening a browser and testing these URLs:


Enter SAP backend gateway credentials.

URL should return a page with this information:

Server reached successfully

7. Testing Fiori URL using Apache with HTTPS connection. Port 443 is the default https port.

**Scenario 3:** In this section, Apache as reverse proxy using two-way HTTPS communication (mutual authentication) is covered in the following steps:

1. Create X.509 client certificate (SAP Gateway)
2. Adding profile parameter for client verification and trusted relationship (SAP Gateway)
3. Changing the Logon Procedure
4. Configure httpd.conf file for mutual authentication (Apache)
5. Restart Apache Server
6. Load.p12 Client Certificate in to the browser
7. Verify two-way mutual communication
8. Testing OData using Apache Server URL (two-way HTTPS protocol)

In two-way SSL, client authenticates the server & the server also authenticates the client, public cert of the Gateway server needs to be configured in the trust store of the Apache server. Also the public cert of the Apache needs to be configured on the Gateway server's trust store. Gateway Server and the Apache must have SSL certificates issued by an authorized certificate authority. An issued certificate includes a digital signature confirming the identities of the gateway server and the Apache Server. When the Apache's host sends a request to the gateway server, the server will verify that the Apache has an SSL certificate and vice versa. There are six steps to achieve this task:

1. Create X.509 client certificate

To digitally identify a particular individual client certificates are used. In general, certificates are issued by company’s PKI (X.509 Public Key Infrastructure). In our case, to test our scenario we will create end user certificates using OpenSSL.

**Step 1:** Downloaded OpenSSL from following link:

http://slproweb.com/download/Win64OpenSSL-1_0_0n.exe

**Step 2:** Generate RSA

Go to command line and CD to path where OpenSSL is installed. For example, C:\OpenSSL-Win64\bin

openssl genrsa -des3 -out server.key 2048

**Step 3:** Create CSR file

openssl req -sha256 -out SUPUSER.csr -new -newkey rsa:2048 -nodes -keyout server.key

**Step 4:** Sign it using your internal CA

**Step 5:** You will receive signed certificate. Save the SUPUSER.CRT to local drive.

**Step 6:** Convert to crt to pfx format

openssl pkcs12 -export -out SUPUSER.pfx -inkey server.key -in SUPUSER.crt

Note: Enter password if required.
Step 7: Maintain User mapping

TCode: go to SE11,
Database Table: VUSREXTID and hit Display
Click on Contents (CtI+sht+f10)
External ID Type: DN of certificate (DN)
Click and change and New entries
- External ID should be the Subject DN
- Assign the user (prior to this activity make sure user is already created)
- Check Activated

Important: Subject DN should match with the VUSREXTID entry as shown in the following screens:
2. Adding profile parameters for client verification and establishing trusted relationship between Apache and ICM

Step 1: TCode: RZ10

Set the AS ABAP profile parameter icm/HTTPS/verify_client to the value 1 (accept certificates) or 2 (require certificates) to support the use of client certificates.

Step 2: TCode: RZ10

For X.509-based logon to NW AS using the Apache Server, you need following parameters to create a trusted relationship between the Apache Server and ICM

icm/HTTPS/trust_client_with_issuer = <Root Subject DN of the Apache Server>
icm/HTTPS/trust_client_with_subject = <Host Subject DN of the Apache Server>

Example:
icm/HTTPS/trust_client_with_issuer = EMAIL=maik.mueller@sap-ag.de, CN=SAPNetCA, OU=SAPNet, O=SAP-AG, C=DE
icm/HTTPS/trust_client_with_subject = CN=USPHLVM1383.PHL.SAP.CORP, OU=COE, O=SAP-AG, C=DE

3. Changing the Logon Procedure

Step 1: TCode SICF
Navigate to default_host/sap/ui5_ui5/ui2/
Service Name: ushell
Change Logon Procedure to: Required with SSL Certificate
Step 2: Restart ICM

Step 3: Testing SAP Gateway Fiori URL

*Note: please refer Point 5 for loading p.12 certificate into browser*


When you hit the Gateway URL, a popup for certificate selection is appeared. You select the right end user certificate and hit OK.

![Select a certificate](https://mo-06a18f128.mo.sap.corp:8400/sap/bc/ui5_ui5/ui2/ushell/shells/abap/FioriLaunchpad.html?sap-client=100&sap-language=EN)

User is authenticated with X.509 Certificate as shown below:

![User authenticated](https://mo-06a18f128.mo.sap.corp:8400/sap/bc/ui5_ui5/ui2/ushell/shells/abap/FioriLaunchpad.html?sap-client=100&sap-language=EN)

```
Now next step is doing the same via Apache Server.

4. Adjust the httpd.conf file for mutual authentication (Apache Server)

SSLProxyMachineCertificateFile used in httpd.conf MUST be in PEM format. You can use openssl for conversion by running below commends for your server (ApacheServer.crt) and root certificate (SAPNetCA.crt).

   a) openssl x509 -in ApacheServer.crt -out ApacheServer.der -outform DER
   b) openssl x509 -in ApacheServer.der -inform DER -out ApacheServer.pem -outform PEM
```
c) openssl x509 -in SAPNetCA.crt -outform DER
d) openssl x509 -in SAPNetCA.der -inform DER -out SAPNetCA.pem -outform PEM

NOTE: If server or root certificate is in the .der format then you can use b) or d) option to convert into PEM format

SSLProxyMachineCertificateFile - point it to a file containing your Apache server certificate which is converted into ApacheServer.pem format and its (unencrypted) private key (server.key) in PEM format. (For example, add server.key to ApacheServer.pem). Apache won't start if this is not done correctly. Following the same screen:

In the following example, Apache Server https://usphlv1383.phl.sap.corp:8443/ is mapped to following SAP Gateway Node:

- mo-06a181f28.mo.sap.corp:8400

Listen 8443
<VirtualHost *:8443>
ServerName usphlv1383.phl.sap.corp
SSLEngine On
SSLProxyEngine On
ProxyRequests Off
ProxyPreserveHost On
SSLProxyCheckPeerCN off
SSLProxyCheckPeerName off
SSLVerifyClient require
RequestHeader unset Accept-Encoding
SSLVerifyDepth 10
SSLCertificateFile /Apache24/conf/ApacheServer.crt
SSLCertificateKeyFile /Apache24/conf/server.key
SSLCertificateFile /Apache24/conf/crts/SAPNetCA.pem
SSLProxyCACertificateFile /Apache24/conf/crts/SAPNetCA.pem
SSLProxyMachineCertificateFile /Apache24/conf/ApacheServer.pem
ProxyPassReverseCookiePath / /
ProxyHTMLEnable On
AllowEncodedSlashes On
ProxyPass / https://mo-06a18f128.mo.sap.corp:8400/ nocanon
ProxyPassReverse / https://mo-06a18f128.mo.sap.corp:8400/
RequestHeader set SSL_CLIENT_CERT ""
RequestHeader set SSL_CLIENT_CERT "%(SSL_CLIENT_CERT)s"

CustomLog "c:/Apache24/logs/ssl_request__LB_8082.log" "%t %h %r %s %l %p User:%u %{Foobar}i client_cert:%{SSL_CLIENT_CERT}x client_verify:%{SSL_CLIENT_VERIFY}x client_cert_dn:%{SSL_CLIENT_S_DN}x ""%r" %b"
</VirtualHost>

5. Restart the Apache Server

6. Load .p12 Client Certificate in to the browser

For mutual authentication using client certificates, Apache needs the private keys to do the signing, and the .p12 file format is the most common for passing around a certificate with its private keys. To test, we need client certificate (.p12 file) which is usually provided by your OS security team who handles Certificate Authority.

1. Load the .p12 client certificate into the personal certificate store. In Chrome, choose Settings > Show Advanced Settings > HTTPS/SSL > Manage certificates as shown below screen:

2. Click Import button:
3. Click Next button:

4. Click browse and select the p.12 file
5. Select All files from dropdown:

6. Select p.12 and hit Next button as shown below:
7. If password exists, provide password and hit next:

NOTE: s_client is a diagnostic tool for OpenSSL. For more information, refer following link
https://www.openssl.org/docs/apps/s_client.html

Example for testing client certificates:

C:\OpenSSL-Win32\bin>openssl s_client -connect usphlv1384.phl.sap.corp:8443

7. Verify two-way HTTPS Scenario

Validate the configuration by opening a browser and testing these URLs:


When you hit the Apache URL, a popup for certificate selection is appeared. You select the right end user certificate and hit OK.

URL should return a page with the below information:

Server reached successfully

8. Testing Fiori URL using Apache with mutual HTTPS connection.

Apache 8443 result Logs:

User: -  client_cert:-----BEGIN CERTIFICATE-----
MIICyzCCAjNgAwIBAgIEAQB77DANBgkqhkiG9w0BAQUFADBpMQswCQYDVQQGEwJE
RTEPMA0GA1UdMGMGCCsGA1UMBIGA1UdMGMGCCsGA1UMBIGA1UdMGMGCCsGA1UMBIGA1
UE5ldENBMSUwIwYJKoZIhvcNAQEFBQADgYEAphhpAQ08褛oneZlXkGx+SM71ZhoY
mpnFlxVxRggbzIcLbObt2E4/EK9tqTet4aI486H2dZNYg+f3hbcFxNGOOfk
vodCeMrZ7uKJdGhOxXerOfRDKy5H32XXxXVnSB/odRoFMEvEVm9IoUyYvr6M4=
-----END CERTIFICATE-----
client_verify:SUCCESS client_cert_dn:CN=SUPUSER,OU=SSL Server,O=SAP-AG,C=DE

Next section covers Apache server monitoring aspects.
6. Monitoring settings for Apache Server

In this section, we will cover monitoring aspects.

Server Status:

The Status module allows a server administrator to find out how well the server is performing. A HTML page is presented that gives the current server statistics in an easily readable form. If required this page can be made to automatically refresh (given a compatible browser). Another page gives a simple machine-readable list of the current server state.

We can setup a password for accessing these details. Example for setting up basic authentication:

```
C:\Apache24\bin>htpasswd -c /Apache24/conf/passwd-server-status kkola
New password: **********
Re-type new password: **********
Adding password for user kkola
```

Example for server status configuration in `httpd.conf` file:

```
<Location /server-status>
    SetHandler server-status
    Order Deny,Allow
    Deny from none
    Allow from all
    AuthType basic
    AuthName "Apache server-status"
    AuthUserFile /Apache24/conf/passwd-server-status
    Require valid-user
</Location>
```

URL to access: http://hostname:port/server-status
In addition to above features, there are plenty of open source tools available to monitor and manage Apache Servers.

In the next section, we will focus on the some troubleshooting techniques for communication issues.
Troubleshooting

7. Troubleshooting communication issues

**Issue 1:** 404 error code related issues

Solution:

To pass the URL path "raw" to the backend use `nocanon` at the end of `ProxyPass` directive.

```
ProxyPass /yourpath http://server.domain:port/yourpath nocanon
ProxyPassReverse /yourpath http://server.domain:port/yourpath
```

**Issue 2:** Encoded slashes when using Apache as proxy results in 404 errors

Solution:

The Apache directive `AllowEncodedSlashes=ON` may be used.

(http://httpd.apache.org/docs/2.2/mod/core.html#allowencodedslashes)

**Issue 3:** X.509 client certificate authentication issues

Solution:

Example Scenario:

Backend is configured to use X.509 certificates to authenticate Fiori application but once certificate is presented basic authentication screen is displayed.

Step 1: Increase your ICM trace to level 2 for retrieving detailed SSL debug information. For more information refer following link:

http://help.sap.com/saphelp_nw70ehp2/helpdata/en/48/3a062c902131c3e10000000a42189d/content.htm

In my case, I see following error in ICM trace:

```
[Thr 100183722298496] HttpSubHandlerCall: Call Handler: HttpModHandler (132a586/13c7409), task=TASK_REQUEST(1), header_len=1581
[Thr 100183722298496] <- SapSSLGetPeerInfo(ssl_hdl=12ab000)==SAP_O_K
[Thr 100183722298496] | out: subject = "CN=IDP5HLVM103.PHL.SAP.CORP, O=IDP, C=IN, C=US, C=US"
[Thr 100183722298496] | out: issuer = "EMAIL=mail.mueller@ap-sg.de, CN=SAPNetC3, O=SAPNet, C=SAP-AG, C=DE"
[Thr 100183722298496] | out: cert_len = 661
[Thr 100183722298496] | out: cipher = "TLS_RSA_WITH_AES128_CBC_SHA"
[Thr 100183722298496] [Thr 100183722298496] HttpModGetDefRules: Client certificate received: with len=661, subj="CN=IDP5HLVM103.PHL.SAP.CORP, O=IDP, C=IN, C=US, C=US"
[Thr 100183722298496] [Thr 100183722298496] HttpModGetDefRules: Intermediate cert subject not trustworthy: "CN=IDP5HLVM103.PHL.SAP.CORP, O=IDP, C=IN, C=US, C=SAP-AG"
[Thr 100183722298496] [Thr 100183722298496] HttpModGetDefRules: profile trust_client_with_subject: "CN=usphlvm103.phl.sap.corp, O=SSL_ Server, C=SAP-AG"
[Thr 100183722298496] [Thr 100183722298496] BINDING of content denied
[Thr 100183722298496] [Thr 100183722298496] HTTP request: (2/12/21) Revert untrusted forwarded certificate (received via HTTPS with untrusted certificate)
[Thr 100183722298496] [Thr 100183722298496] HttpModDefRules: determined the definitions: REMOVE_SSL_HEADER (8)
[Thr 100183722298496] [Thr 100183722298496] HttpModHandler: remove incoming ssl header
[Thr 100183722298496] [Thr 100183722298496] HttpModHandler: perform the actions: REMOVE_SSL_HEADER (8)
[Thr 100183722298496] [Thr 100183722298496] http:://ihttp:9443传奇私服 -> 17f4200 687356 (0) -> 7f68557c3328 1e455769b0 htp::OK
[Thr 100183722298496] [Thr 100183722298496] HttpModHandler: serialize new http header
[Thr 100183722298496] [Thr 100183722298496] IOT: IOTRequestMessage( 137e80 ) -> w=0 cmd=0
[Thr 100183722298496] [Thr 100183722298496] IOT: IOTRequestMessage( 137e80 ) -> w=0 cmd=0
[Thr 100183722298496] [Thr 100183722298496] BINDING of content denied
```
Above error is related to certificate subject. Next step is to verify the profile parameters:

From the profile, we can see missing C=DE in the client subject. Then, subject DN is modified and restarted ICM. After retesting the application successfully, following “Accept trusted forward certificates” message is displayed in ICM trace.
In summary, this white paper covers how to proxy Fiori applications via Apache Server. Plain http, one-way https, and X.509 mutual authentication steps are covered.