SESUG Paper LRN8-2023

Debugging Tips for PROC HTTP

Kim Wilson, SAS Institute Inc.

ABSTRACT

Several great papers have been written about how to get started with PROC HTTP, which includes accessing Microsoft 365 applications, modifying various options for desired results, and more. As a SAS Technical Support Engineer, I often assist SAS customers who are not receiving the expected resource, or they are seeing a return code that is not a 200 OK. This paper describes common errors that you might encounter regarding certificates, authentication, and general errors, as well as overall debugging techniques and suggestions. This paper also helps you gather pertinent information that SAS Technical Support will need when helping to solve the problems occurring with or around PROC HTTP.

INTRODUCTION

We all interact with websites on a regular basis by specifying the web address, or URL, whenever we want to know sports scores, check the latest news headlines, or see a restaurant's menu…and so much more. When accessing those websites, the browser interacts with a server, and HTTP is the foundation for the internet that retrieves the requested information. Many web applications are widely used today, and web services use HTTP for communication between the client and the server with various interfaces, such as through a REST API. PROC HTTP allows you to access application programming interfaces (APIs) with SAS® 9.4 and SAS® Viya®. The log displays the return code from PROC HTTP, and a 200 OK status means the request was successful. This paper gives tips and suggestions when your return code is not 200.

HTTPS refers to secure HTTP protocol. Although it functions similarly to HTTP, HTTPS is more secure and uses TLS/SSL to encrypt and decrypt normal HTTP requests and responses.

Note: This paper contains some information about the Microsoft Windows and z/OS operating systems, but it most often uses examples specific to the UNIX operating system.

THINGS TO CONSIDER BEFORE CONTACTING SAS TECHNICAL SUPPORT

As you encounter unexpected results or errors from PROC HTTP code, these are a few tips to help you start your troubleshooting efforts before contacting Technical Support:

- Run `%PUT _AUTOMATIC_;` in your SAS session to write the values of many helpful macro variables to the log:
  - &SYSHOSTNAME shows the machine where SAS is executing. If executed on the SAS Viya platform, SYSHOSTNAME and SYSTCPIPHOSTNAME return the name of the pod.
  - &SYSSCPL displays the name of the operating environment.
  - &SYSVLONG shows the release of SAS.
- Add the DEBUG statement with the LEVEL= option to your PROC HTTP code so that debugging information is written to the SAS log.
- Determine the authentication method in use for the API from your API documentation.
- Are you able to reach the API from the machine executing SAS?
  - Try using cURL (or another system tool) from a command line on the machine executing SAS.
  - Try to access a free HTTP request and response testing service such as HTTPBIN.ORG as discussed in The ABCs of PROC HTTP.

- For any endpoint that you want to reach with PROC HTTP, SAS can do only what is technically possible from the machine executing SAS. If the SAS server is unable to connect to the desired URL, PROC HTTP will not be able to either.

MICROSOFT 365 APPLICATIONS (ONEDRIVE, TEAMS, SHAREPOINT)

Customers often contact SAS Technical Support about how to read and update Microsoft applications. Reference SAS KB0036234 for possible code strategies to access SharePoint from SAS depending upon your SAS version, operating system, SharePoint version, and SharePoint authentication setting.

Chris Hemedinger wrote Using SAS with Microsoft 365 (OneDrive, Teams, and SharePoint) and recorded a video of the step-by-step process for accessing Microsoft 365 applications. It is often helpful to watch each step of the video and then pause it while you perform the same step. If you're still unable to make the connections Chris describes, consult your IT infrastructure team to verify that you are able to connect to the appropriate resources. If you have difficulties implementing Chris' code, SAS Technical Support will ask for proof that you can access the APIs from outside of SAS. If you are not able to access the Microsoft resources with PROC HTTP, we suggest that you attempt to access them with cURL or Postman before contacting Technical Support. Then send us the successful cURL command. The How to translate your cURL command into SAS code blog is helpful with the conversion to PROC HTTP code.

If you have SharePoint 2017 with NTLM authentication in place and would like to access files on the SharePoint site, an AUTH_NTLM argument is available for PROC HTTP. But note that it is ONLY available for SAS on Microsoft Windows. To access the SharePoint site from SAS executing on UNIX, you should work with your SharePoint team to determine a cURL method to access the SharePoint site. Once you have cURL command(s) that obtain successful results from the command line of the machine where SAS is executing, then you can use the SYSEXEC statement in the DATA step to submit the cURL commands from SAS. You'll need to ensure that the XCMD command is enabled in order to use the SYSEXEC command from the DATA step. You can check the XCMD value from SAS with this code:

```
proc options option=xcmd;
run;
```

TLS/SSL ISSUES

SAS®9 uses OpenSSL to interact with TLS on UNIX and z/OS operating systems, and Windows uses the Schannel.

Beginning in SAS® 9.4M3, the Mozilla bundle of trusted certificate authorities (CA) certificates are shipped with the SAS install for the UNIX and z/OS operating systems, and the system option SSLCALISTLOC is set to specify the location of these trusted CA certificates. Starting with SAS 9.4M3, the default path set for the SSLCALISTLOC= SAS system option on the UNIX and z/OS foundation servers is <SASHome>/SASSecurityCertificateFramework/1.1/cacerts/ trustedcerts.pem.

The purpose of shipping the trustedcerts.pem file is to allow customers access to all the common CA certificates that common web servers use, so that they do not have to configure all these common CA certificates. See the code in Table 1 for how to access them:

Table 1. Certificate Location by SAS Platform

	SAS® 9.4	**SAS® Viya® 3.5**	**SAS® Viya® platform**
Location of certificates on Unix and z/OS	Proc options option=sslcalistloc;	Proc options option=sslcalistloc;	%put %sysget(SSL_CERT_FILE);

To obtain the location of the Schannel interface with the Windows CA store, run the **mmc** Windows command. See Configure TLS and Request Digital Certificates on Windows for more information about certificates on Windows.

If you need to verify the format of the trustedcerts.pem or any custom certificates, you can use a BASE64 decoder/encoder similar to the following when comparing your certificates with those in the SAS .pem file: https://www.sslshopper.com/certificate-decoder.html.

Foundation SAS® components using TLS/SSL require a trusted connection, so the SAS session needs to have access to the web server's CA certificate. The web server's public certificates cannot be self-signed or, if they are self-signed, Basic Constraints should be set to **true**.

When errors appear in the SAS log that indicate problems with SSL, it's helpful to obtain enhanced logging to investigate. If you can replicate the error or issue from a Base SAS® session (batch, Display Manager System (DMS), or line mode), you can use the %log4sas macros to enable enhanced logging. Here is an example of suggested code for you to run and then examine the resulting log:

```
 %put _automatic_;

%log4sas();
%log4sas_logger('App.tk.eam.ssl', 'level=trace '); run;
%log4sas_logger('App.tk.HTTPC', 'level=debug'); run;
%log4sas_logger('HTTP', 'level=trace'); run;
%log4sas_logger('Perf', 'level=error'); run;
%log4sas_logger('App', 'level=debug'); run;
%log4sas_logger('App.Program', 'level=trace '); run;
%log4sas_logger('Audit.Authentication', 'level=error'); run;

<your SAS PROC HTTP code goes here >
```

If you are unable to run code from a Base SAS session, you can follow the directions in SAS Note 63587 to generate a debug log separate from the default SAS log. Be aware that this logging affects all users who are executing code that writes logs to the Workspace Server, which will fill the logs very quickly. Consider executing the code when the fewest users are executing jobs.

In SAS® 9.4M8, changes occurred with the SSL/TLS interface on UNIX and z/OS. SAS no longer ships the OpenSSL libraries on UNIX and z/OS. UNIX uses the system OpenSSL libraries, and z/OS uses IBM System SSL. SAS Note 69954 describes more z/OS changes for SAS 9.4M8. Details about encryption in SAS 9.4 can be found in the What's New for Encryption documentation.

Errors similar to these might occur, indicating that the SAS OpenSSL libraries could be newer than those that the remote web server is using. These errors are not unique to SAS, and you can find additional details by searching the web.

```
ERROR: OpenSSL error 167772498 (0xa000152) occurred in SSL_connect/accept
at line 5109, the error message is "error:0A000152:SSL
        routines::unsafe legacy renegotiation disabled".
ERROR: Secure communications error status 807ff008 description
"34.197.235.101: OpenSSL error 167772498 (0xa000152) occurred in
        SSL_connect/accept at line 5109, the error message is
"error:0A000152:SSL routines::unsafe legacy renegotiation disabled".
```

To address these errors, you might need to adjust some variables. For more information, see Set Library Path Variables to OpenSSL Libraries in SAS 9.4M8 in the *SAS® 9.4 Administration* documentation.

One useful option available in SAS® 9.4M5 is SSLMINPROTOCOL, and the default value is TLS 1.2. Starting with SAS 9.4M8, TLS 1.3 is now supported and can be set as a minimum value. Before SAS 9.4M5, many SSL options were set as environment variables.

You can use a tool, such as OpenSSL, to verify the TLS version of the remote web server:

```
openssl s_client -connect web_server_name:443
```

This code allows you to see the values of numerous system options related to networking and encryption:

```
Proc options group=communications;
Run;
```

COMMON CERTIFICATE ERRORS

This error might occur in your SAS log:

```
ERROR: OpenSSL error 336134278 (0x14090086) occurred in  SSL_connect/accept
at line 6274, the error message is "error:14090086:SSL
routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed"."
```

This error indicates that the connection is not trusted. It might have failed to verify because the CA Root certificate (or intermediate CA certificate) was expired. In this scenario, a message similar to the following appears in the enhanced logging:

```
TLS certificate verification: Error, certificate has expired
```

Other possible causes are due to a missing intermediate CA certificate, CA root or intermediate certificates being added in the wrong order and not being found, or a certificate in the chain being self-signed.

You add the unexpired certificates to your SAS 9.4 deployment using the SAS Deployment Manager by following the instructions in these links:

- For SAS 9.4: Manage Certificates in the Trusted CA Bundle Using the SAS Deployment Manager
- For the SAS Viya platform: How To

Here is a common set of errors related to TLS/SSL "`certificate verify failed`" message that indicates the connection is not trusted and failed:

```
ERROR: OpenSSL error 336134278 (0x14090086) occurred in SSL_connect/accept
at line 6276, the error message is "error:14090086:SSL
routines:ssl3_get_server_certificate:certificate verify failed.
ERROR: Encryption run-time execution error
ERROR: Secure communications error status 807ff008 description
"nnn.nnn.nnn.nnn: OpenSSL error 336134278 (0x14090086) occurred in
SSL_connect/accept at line 6276, the error message is "error:14090086:SSL
routines:ssl3_get_server_certificate:certificate
ERROR: Encryption run-time execution error
```

To begin debugging why the connection wasn't trusted, there are several choices. You can determine the value of &SYSHOSTNAME as covered earlier, and then test the connection outside of SAS using a utility such as cURL executed from the command line of the machine where SAS is running:

```
curl -v 'https://<your_url>' -o /tmp/out.txt
```

If this cURL command is successful, you can compare the cURL verbose output with the enhanced logging, as discussed above. You can also compare the certificates used in both environments. To see the location of the CA certificates used by SAS, execute the following command:

```
proc options option=sslcalistloc;
run;
```

If the cURL command fails, then you need to engage your IT resources to determine which certificates are needed to make this connection.

You can also run this OpenSSL command, which writes the contents of your certificates to a file named ms.pem, as shown in this example of connecting to graph.microsoft.com:

```
echo -n | openssl s_client -showcerts -connect graph.microsoft.com:443 |
sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /tmp/ms.pem
```

Note: The -showcerts option shows the full certificate chain.

Then compare the contents of ms.pem file with the contents of the trustedcerts.pem file.

On the SAS Viya platform, you can copy the trustedcerts.pem contents with this command, where `podname` refers to a pod such as sas-compute-server-NN or sas-studio-app:

```
kubectl cp podname:/security/trustedcerts.pem /tmp/trustedcerts.pem
```

To show the names of the pods, run the following command:

```
kubectl -n namespace get pods
```

Sample output is shown below:

Then, run this code, inserting the name of the pod from above into the `podname` placeholder:

```
kubectl -n namespace cp sas-studio-app-8b597c597-
q4kd8:/security/trustedcerts.pem /tmp/studio.pem
```

Note that /security/trustedcerts.pem file is built dynamically by sas-certframe, so its date modified would be the date and time that the pod started. You can look at the trustedcerts.pem if you exec into a pod, similar to this:

```
kubectl -n namespace exec sas-studio-app-8b597c597-q4kd8 -it -- bash
```

This command shows the contents of the **`/tmp`** directory, and you can look at the date and time of the .pem file.

```
ls -lt /tmp
```

If you want more information, see the SAS certificate documentation for the SAS Viya platform: SAS Help Center: How To

NEGOTIATIONS AND CIPHERS

Early SAS 9.4 releases might need to apply SAS SSL security vulnerability hot fixes. This link takes you to the SAS Statement Regarding OpenSSL Security Advisories.

These hot fixes not only address security vulnerability issues, but they also make the environment variable SAS-SSL_MIN_PROTOCOL available in SAS 9.4M0 - SAS 9.4M2. On UNIX and z/OS, the OpenSSL libraries are upgraded with these hot fixes.

After the necessary certificates are verified, the client SAS session (PROC HTTP) and the web server must negotiate which TLS components can be used in the TLS/SSL connection. For example, SAS and the web server will negotiate the TLS release and the ciphers used.

The version of OpenSSL libraries used determines which versions of TLS/SSL are enabled and/or disabled. The version of TLS/SSL also affects which ciphers are available to use in negotiations, along with values used for the SAS options SSLMODE= and SSLCIPHERLIST=. The SSLMODE= option and available ciphers are discussed in the SSLMODE= System Option documentation. Also see CipherSuites in TLS/SSL (Schannel SSP).

A cipher suite is a set of algorithms that are used to secure a connection via the TLS or SSL protocols between clients and servers. The negotiated ciphers are also affected by the algorithm used in the creation of any certificates. The ciphers format consists of the following:

- A key exchange algorithm
- An authentication algorithm
- Bulk Data Encryption
- A message authentication code (MAC) algorithm

If the negotiated cipher's authentication algorithm uses ECDSA, then all certificates in the chain must also use ECDSA. The CA certificates in the SAS shipped Mozilla trustedcerts.pem use the RSA authentication algorithm and will not work with ciphers using the ECDSA authentication algorithm.

Even if available ciphers used by SAS and the web server are negotiated successfully, if the CA certificate and public certificate were created using different authentication algorithms than the negotiated cipher, the connection will fail. To address this situation, you can limit the ciphers available with the option SSLCIPHERLIST. Sometimes, the client and server are not able to agree upon the ciphers, which could lead to errors such as these:

```
ERROR: Secure communications error status 807ff03e description
"nn.nnn.nn.nnn: A handshake failure has occurred. This may be because the
remote host cannot support the cipher suites required by the SSL mode."
ERROR: A handshake failure has occurred. This may be because the remote
host cannot support the cipher suites required by the SSL mode.
ERROR: Call to tcpSockContinueSSL failed.
```

You can compare the verbose output from a successful cURL command with the SAS enhanced logging to debug any cipher issues.

If running SAS Viya 3.5, see Problem Note 68586, "SAS® clients fail to connect, and you see an 'error:1408A0C1:SSL routines:ssl3_get_client_hello:no shared cipher' message in server logs."

There are many other reasons that you might encounter errors related to certificates. If you have difficulties interpreting the enhanced logging, Technical Support is available to help investigate.

RETURN CODES

400 - BAD REQUEST

What it means: You formatted something incorrectly to the endpoint, such as a problem with syntax in PROC HTTP. For example, you asked for A and you needed to ask for a, you put {} and you needed [], or you requested a menu from the HR department instead of from the café.

Where to look for answers: Compare the results from the DEBUG statement (set to level 2) with verbose cURL results. If the cURL command returns a 400 return code, then you need to work with your infrastructure team or consult your API documentation in order to create a valid request. If you have difficulties translating a successful cURL command to PROC HTTP code, consult the How to translate your cURL command into SAS code blog or contact SAS Technical Support.

401 - UNAUTHORIZED

What it means: The URL is valid, but the user or user ID that is making the request is not authorized for this action. The request has not been applied because it lacks valid authentication credentials for the target resource.

Where to look for answers: Use the cURL command to see whether you are successful reaching the endpoint.

Here is an example:

When you submit PROC HTTP code to access SharePoint, the defined authentication method is NTLM. You are running on UNIX, so a 401 Unauthorized message could appear in the log. See The ABCs of PROC HTTP for more information about authentication.

403 - FORBIDDEN

What it means: There is an issue with the user ID and credentials. The endpoint can identify and recognize that you do not have permissions for the requested location. The authentication was provided, but the authenticated user is not permitted to perform the requested operation.

Where to look for answers: Make sure that you have the right permissions to access the endpoint. Use cURL to compare, and if cURL return code matches the return code from PROC HTTP, consult with your infrastructure team.

Here is an example:

When you run PROC HTTP code consecutively and encounter return code 403, you might need to add the CLEAR_CACHE option to the PROC HTTP code.

Persistent connections, or HTTP keep-alive, is a way to send and receive multiple requests/responses using the same connection. This is used extensively in web browsers because it can reduce latency tremendously by not continually creating new connections, which reduces the overhead of TLS handshakes. Starting in SAS 9.4M3, PROC HTTP uses persistent connections. Connections are kept alive by default, but there are various ways to disable or close a connection if needed. The CLEAR_CACHE option specifies to clear both the shared connection and cookie caches before the HTTP request is executed.

502 - BAD GATEWAY

What it means: There is a problem with the connection to the URL that PROC HTTP is attempting to access, perhaps with the proxy or network. It does not imply that there is a problem with the PROC HTTP syntax.

Where to look for answers: Note the times of the failures and ask your systems or security team to help set up debugging for the next time the problem occurs. For the URL that returned the 502, it is a good idea to examine the access or error logs to determine the cause of the failure.

As stated before, when you encounter a non-200 return code from PROC HTTP, you should use a system utility such as cURL to see if the return code is replicated.

PROXY-RELATED ISSUES

There are several types of errors that suggest something is happening related to a proxy server. For example, this error might appear in your log:

```
ERROR: Error connecting to nn.nnn.nn.nn:443. (The connection has timed
out.)
```

In this scenario, contact your systems team to ask whether a proxy is required for you to make the external URL accessible from the machine that is running SAS. You can add the proxy information to your PROC HTTP code with the PROXYHOST= and PROXYPORT= options.

Here is another set of errors that you might encounter:

```
ERROR: Unable to establish connection to servername.org via proxy
nnn.nn.nnn.nnn.
ERROR: Unable to connect to Web server.
```

In this scenario, see How to secure your REST API credentials in SAS programs.

Here is an example of adding proxy, web user name, and password:

```
proc http url = &url.
          method=GET
          out=resp
          proxyhost="nnn.nn.nnn.nnn"
          proxyport=8080
          WEBUSERNAME="sa_proxy_datalab"
          WEBPASSWORD=XXXXXXXXXXXXXXXX;
          headers  "Ocp-Apim-Subscription-Key" = "&subscriptionkey."; ;
run;
```

Errors similar to these might occur in your log:

```
ERROR: Error connecting to nn.nn.nnn.nnn:443. (The connection was refused.)
ERROR: Unable to establish connection to server-name.com.
ERROR: Unable to connect to Web server.
```

In this scenario, test with a public testing site to see whether you are able to reach the site without errors using code similar to the following:

```
filename out TEMP;
filename hdrs TEMP;
proc http
url="https://httpbin.org/get"
method="GET"
out=out
headerout=hdrs;
run;
```

If you are still receiving errors, it is likely that you need a proxy to get access to HTTPS sites from your machine where SAS is running. Consult with your systems team to determine the correct proxy host and port values.

ADDITIONAL SAS VIYA PLATFORM INFORMATION

The SAS Viya platform contains many APIs, which can be accessed using PROC HTTP.

See the following resources for more information:

- Authentication to SAS Viya: a couple of approaches blog by Joe Furbee
- SAS Viya REST APIs documentation on developer.sas.com.
- https://developer.sas.com/apis/rest/SAS Viya REST APIs.

CONCLUSION

PROC HTTP is widely used to interact with web services. When APIs are involved, it is imperative that you prove access to the API in question from the machine where SAS is running. You can find this server name by submitting this code: `%put &SYSHOSTNAME;`. Once you can communicate with the API from the SAS server outside of SAS by using cURL or another method, you should be able to use PROC HTTP to complete the same task from within SAS.

REFERENCES

Furbee, Joe. 2023. "Authentication to SAS Viya: a couple of approaches." Available at **blogs.sas.com/content/sgf/2023/02/07/authentication-to-sas-viya/**.

Henry, Joseph. 2020. "REST Just Got Easy with SAS® and PROC HTTP." *Proceedings of the SAS Global 2020 Conference*, Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings20/4426-2020.pdf**

Henry, Joseph. 2019. "The ABCs of PROC HTTP." *Proceedings of the SAS Global 2019 Conference*, Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings19/3232-2019.pdf**

Hemedinger, Chris. 2018. "How to secure your REST API credentials in SAS programs." Available at **blogs.sas.com/content/sasdummy/2018/01/16/hide-rest-api-tokens/**

Hemedinger, Chris. 2020. "Using SAS with Microsoft 365 (One Drive, Teams, and SharePoint)." Available at **blogs.sas.com/content/sasdummy/2020/07/09/sas-programming-office-365-onedrive/**

Lawhorn, Bari. 2020. "How to translate your cURL command into SAS code." Available at **blogs.sas.com/content/sgf/2020/07/30/curl-to-proc-http/**

SAS Institute Inc. 2023. "Configure TLS and Request Digital Certificates on Windows." *SAS® 9.4 Administration*. Cary, NC: SAS Institute Inc. Available at **go.documentation.sas.com/doc/en/bicdc/9.4/secref/p1g2v5c010q6gyn1fi8f17r7pdi3.htm**.

SAS Institute Inc. 2023. "How To." *SAS® Viya® Platform Administration*. Cary, NC: SAS Institute Inc. Available at **go.documentation.sas.com/doc/en/sasadmincdc/v_044/calencryptmotion/n1xdqv1sezyrahn17erzcunxwix9.htm**.

SAS Institute Inc. 2023. "Incorporate Additional CA Certificates." *SAS® Viya® Platform Administration*. Cary, NC: SAS Institute Inc. Available at **go.documentation.sas.com/doc/en/sasadmincdc/v_038/calencryptmotion/n1xdqv1sezyrahn17erzcunxwix9.htm#n00kfxh05vul45n1ki9k5x6jj4bc**.

SAS Institute Inc. SAS KB0036234. "How to write to or read from a Microsoft SharePoint site using SAS® software." Available at **sas.service-now.com/csm?id=kb_article_view&sysparm_article=KB0036234**.

SAS Institute Inc. 2023. “Manage Certificates in the Trusted CA Bundle Using the SAS Deployment Manager.” *SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation*. Cary, NC: SAS Institute Inc. Available at **go.documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/secref/n0n1y5gwevy312n13h5bm4yf6quy.htm**.

SAS Institute Inc. SAS Note 63587. “Obtaining additional debugging log information for the HTTP procedure.” Available at **support.sas.com/kb/63/587.html**.

SAS Institute Inc. SAS Note 68586. “SAS® clients fail to connect, and you see an ‘error:1408A0C1:SSL routines:ssl3_get_client_hello:no shared cipher’ message in server logs.” Available at **support.sas.com/kb/68/586.html**.

SAS Institute Inc. 2023. “SAS Viya REST APIs.” Available at **developer.sas.com/apis/rest/**.

SAS Institute Inc. 2023. “Set Library Path Variables to OpenSSL Libraries in SAS 9.4M8.” *SAS® 9.4 Administration*. Cary, NC: SAS Institute Inc. Available at **go.documentation.sas.com/doc/zh-TW/bicdc/9.4/secref/p0b8umfste7jvan184a4infhc49q.htm**.

ACKNOWLEDGMENTS

I would like to thank Bari Lawhorn and Cindy Matthews for their technical review and input and Rayna Rowell for the exceptional editing.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Kim Wilson
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Email: **support@sas.com**
Web: **support.sas.com/en/support-home.html**