

## Security Concerns and Best Practices for a SAS Grid

The following white paper will identify security issues and the best practices for addressing security issues when accessing nodes in a SAS grid using the capabilities of SAS/CONNECT. In fact, these same issues apply for a CONNECT client connecting to and accessing a single remote machine as well as a CONNECT client connecting to and accessing many remote machines in a grid. The information presented here applies primarily to using a Windows platform as the remote server. The information will eventually be expanded to include other server platforms as well.

### ***The following are security concerns for using SAS/CONNECT:***

- Password storage and communication to the remote host
- Unauthorized access to remote machines
- Ability to use Security Support Provider Interface (SSPI) to bypass Windows authentication (See the following paragraph for an explanation of SSPI.)
- Ability to run arbitrary commands (X commands) from the server SAS session.
- Risks associated with running the CONNECT spawner
- Ability to secure network transmission of sensitive data.
- Ability to restrict access to other resources on the remote server

SSPI enables transparent authentication for connections between Windows machines. Users that are members of a "trusted" domain are authenticated automatically, and user context information is transferred to the server. Windows attempts to use SSPI for authentication whenever a user ID is not explicitly supplied. SSPI is available **only** when the client and the server sessions both run on Windows machines, **and** the user who runs the client machine is a member of a domain that is "trusted" at the server machine.

The following table summarizes how to run SAS/CONNECT with a high, medium, and low level of security. The sections that follow describe the pros and cons of each level of security and how each of the commands and options enable this level of security.

<b><i>High</i></b>	<b><i>Spawner as service with command line</i></b> <code>spawner -install -netencrypt -netencralg rc2 -noscript -nosspi -sascmd "startsas.bat"</code> <i>contents of startsas.bat</i> <code>sas.exe -noxcmd %*</code> <b><i>SAS/SECURE</i></b> <b><i>PROC PWENCODE</i></b> <b><i>Windows access control</i></b>
--------------------	---

<i>Medium</i>	<p><i>Spawner as service with command line</i></p> <pre>spawner -install -netencralg sasproprietary -sascmd "startsas.bat" contents of startsas.bat sas.exe -noxcmd %*</pre>
<i>Low</i>	<p><i>Spawner as service with command line</i></p> <pre>spawner -install -nosecurity -userid &lt;userid&gt; -password &lt;password&gt;</pre>

## ***Running SAS/CONNECT with a high level of security***

A high level of security protects the transmission of any userid/password values as well as the transmission of any sensitive data across the network. It also prevents unauthorized access to remote servers and the maximum level of control over the SAS processing that gets distributed to any of the remote servers. This prevents unauthorized access and usage of any other resources on the remote servers outside of the SAS environment. This high level of security comes at a higher cost of administration.

In order to run SAS/CONNECT with a high level of security:

1. encryption should be used to secure all network transmissions,
2. the connection should be established using the CONNECT spawner with the spawner running as a windows service and
3. authentication should be required.

In addition, several spawner options related to security should be utilized.

Encryption is available through the use of SAS/SECURE with SAS/CONNECT and is enabled by setting the NETENCRYPT and NETENCALG options on the CONNECT spawner, as shown in the table above, and/or on the remote SAS invocation. You can choose to encrypt just the userid/password credentials during transmission or the credentials as well as all of the data that gets transmitted on the network during the entire connection.

Like the generic telnet daemon, the spawner is a daemon that listens on a specific port for incoming requests for SAS connections. For a high level of security, you should use the spawner instead of the telnet daemon because it allows the administrator of the server machine to completely control the SAS environment that gets created and to provide as much security with that environment as is desired. This is accomplished as follows:

- The spawner should be run as a service by specifying the INSTALL option on the spawner command as shown in the table above. This facilitates administration of the spawner, ensures that the spawner is running with authentication as well as any other options deemed necessary by your administrator, and provides robustness to the environment.

- The spawner should be run in a secured mode which requires user authentication to the server machine in order for the client to even be able to gain access to the server machine in order to invoke SAS. Authentication is on by default if the spawner is run as a service (using the INSTALL option) but off by default if the spawner is not run as a service.
- The spawner supports inheritance by default which means that the spawner listens on a single tcp/ip port and the SAS session that gets created just inherits a socket from this same port for the actual communications of the SAS session. This facilitates the use of a firewall between the client and server machines since only a single port needs to be defined to the firewall.
- The spawner provides various levels of data obfuscation/encryption. By default, the spawner will obfuscate the userid/password as it passes from the client to the server to prevent clear text transmission across the network. Use SAS/SECURE to provide the highest level of security by encrypting the userid/password and all data that flow across the network. This can be done by specifying the NETENCRYPT option on the spawner invocation as shown in the table above. This option requires that encryption be used for all network transmissions and fails connections from any clients that do not support encryption. In addition you should specify the NETENCALG option on the spawner command, as shown in the table above, to provide one or more encryption algorithms to be used to perform the encryption. Note: at least one of the same encryption algorithms must be available at the client as well.
- The NOScript option should be specified on the spawner invocation. This requires that the SASCMD option must be used on the spawner invocation in order to specify the command to invoke the remote SAS session. This allows the administrator of the remote machine to control the options that are specified on the remote SAS batch invocation. We recommend including the NOXCMD option on the SAS invocation which will prevent the client from executing any operating system commands outside of the remote SAS session. It essentially shuts off access to the remote environment and enables execution of SAS program statements only. In order to specify startup options on the SAS invocation, a bat file must be created and the name of the bat file used as the value of the SASCMD option. An example bat file as well as specification of the bat file for the SASCMD option are shown in the table above.
- If both the client and the server machines are windows machines, then it is possible to use SSPI for the authentication and bypass the need for username/password authentication to the remote server. The NOSSPI option has been added to the SAS 9.1 spawner in order to prevent automatic authentication of clients that are members of “trusted” domains. The NOSSPI option is used in the example in the table above. Use of this option does require SAS 9.1.

- In Version 8, the userid and password needed to authenticate the SAS session on the remote machine must either be interactively entered at execution time or stored in a file. Starting in SAS 9.1, PROC PWENCODE can be used to obfuscate the userid and password in a flat file or stored SAS program to prevent clear text storage.
- Having the ability to install a spawner as well as connect to a spawner requires specific operating system rights that must be set up by the administrator. The person who runs the spawner with the –security options must have the following user rights:
  - Act as part of the operating system
  - Bypass traverse checking (the default is everyone)
  - Increase quotas
  - Replace a process level token
  - Log on locally (the default is everyone).

By default, when the spawner is run as a service, it will run as the LocalSystem user which has all of the above rights. Therefore, no additional administration is required.

All users who connect to the spawner must have the “Log on as batch job” advanced right. Therefore, it is typically not possible for a client to just connect to any arbitrary remote pc since it requires that the administrator grant this user right prior to establishing a connection.

- Additionally, a SHELL option is being added to the 9.2 spawner so that execution of operating system commands will be prevented by default by the spawner rather than needing to create a .BAT file to include the SAS invocation with the –NOXCMD SAS startup option..

You could also use access control to better protect the machine. If the spawner is using security, the user's working directory will be set to the user's home folder. Perhaps you could set up special directories for the userid/password being used to connect and allow access only to these directories.

### ***Running SAS/CONNECT with a medium level of security***

A medium level of security provides obfuscation of any userid/password values as well as the transmission of any sensitive data across the network. It also prevents unauthorized access to remote servers and a less stringent level of control over the SAS processing that gets distributed to the remote servers. This helps prevent unauthorized access and usage of any other resources on the remote servers outside of the SAS environment. This medium level of security allows a little more flexibility for the clients

who wish to connect to the remote servers and comes at a lower cost of administration than what a high level of security would involve.

In order to run SAS/CONNECT with a medium level of security:

1. obfuscation should be used to secure all network transmissions rather than the stronger encryption provided by SAS/SECURE,
  2. the connection should be established using the CONNECT spawner with the spawner running as a windows service and
  3. authentication should be required.
- The spawner should be run as a service by specifying the INSTALL option on the spawner command as shown in the table above. This facilitates administration of the spawner, ensures that the spawner is running with authentication as well as any other options deemed necessary by your administrator, and provides robustness to the environment.
  - The spawner should be run in a secured mode which requires user authentication to the server machine in order for the client to even be able to gain access to the server machine in order to invoke SAS. Authentication is on by default if the spawner is run as a service (using the INSTALL option) but off by default if the spawner is not run as a service.
  - The spawner provides various levels of data obfuscation/encryption. For a medium level of security you may not need the stronger encryption of SAS/SECURE but may choose to use obfuscation of the userid/password and/or all of the data that is sent across the network. By default, the spawner will obfuscate the userid/password as it passes from the client to the server to prevent clear text transmission across the network. Use the SASENCALG option with a value of SASPROPRIETARY to provide obfuscation of all data that is exchanged during the entire session. This option is shown in the table above.
  - The SASCMD option should be used on the spawner invocation in order to specify the command to invoke the remote SAS session. This allows the administrator of the remote machine to control the options that are specified on the remote SAS batch invocation. We recommend including the NOXCMD option on the SAS invocation which will prevent the client from executing any operating system commands outside of the remote SAS session. It essentially shuts off access to the remote environment and enables execution of SAS program statements only. In order to specify startup options on the SAS invocation, a bat file must be created and the name of the bat file used as the value of the SASCMD option. An example bat file as well as specification of the bat file for the SASCMD option are shown in the table above. However, be aware that clients connecting with script files would have the ability to override use of the bat file by specifying the SAS invocation in the script file.

- Having the ability to install a spawner as well as connect to a spawner requires specific operating system rights that must be set up by the administrator. The person who runs the spawner with the –security options must have the following user rights:
  - Act as part of the operating system
  - Bypass traverse checking (the default is everyone)
  - Increase quotas
  - Replace a process level token
  - Log on locally (the default is everyone).

By default, when the spawner is run as a service, it will run as the LocalSystem user which has all of the above rights. Therefore, no additional administration is required.

All users who connect to the spawner must have the “Log on as batch job” advanced right. Therefore, it is typically not possible for a client to just connect to any arbitrary remote pc since it requires that the administrator grant this user right prior to establishing a connection

### ***Running SAS/CONNECT with a low level of security***

A low level of security makes the environment easier to configure, easier to maintain, and easier for clients to connect to machines on the network. If you have a trusted environment and set of users, this is the recommended way to run SAS/CONNECT.

In order to run SAS/CONNECT with a low level of security:

1. the connection should be established using the CONNECT spawner with the spawner running as a windows service and
  2. authentication is **not** required.
- The spawner should be run as a service by specifying the INSTALL option on the spawner command as shown in the table above. This facilitates administration of the spawner and provides robustness to the environment.
  - The NOSECURITY option should be used to run the spawner in an unsecured mode. This eliminates the need for the client to authenticate to the server machine in order for the client to connect to the server machine and invoke SAS. Authentication is on by default if the spawner is run as a service (using the INSTALL option) therefore, the NOSECURITY option must be used in order to disable authentication.
  - Use the USERID and PASSWORD options in order to specify a username with which to run the SAS process. This is necessary because a userid/password is not supplied by the client since the spawner is running unsecured. In addition, this

userid must have the “Log on as batch job” advanced right on the remote server because this is the userid associated with the remote SAS process.

- The spawner provides various levels of data obfuscation/encryption. By default, the spawner will obfuscate the userid/password as it passes from the client to the server to prevent clear text transmission across the network. This default behavior is sufficient for a low level of security.

No additional options are recommended for the spawner invocation; sacrificing security in order to provide the greatest degree of flexibility for client connections.