SSL encrypted mode configuration for CAST Storage Service and PostgreSQL

Summary: Instructions for configuring CAST Storage Service/PostgreSQL and CAST AIP related applications to function in SSL encrypted mode.

Introduction

Out of the box, both CAST Storage Service and PostgreSQL are not preconfigured to function in SSL encrypted mode, i.e. to accept incoming encrypted database connections for enhanced security. However, SSL encrypted mode can be configured if required. The configuration process involves specific steps described in this document:

- Generate SSL certificates/keys
- Configure CAST Storage Service/PostgreSQL to accept incoming SSL connections
- Configure CAST AIP core and related applications to function in SSL mode
- Configure CAST AIP Console/AIP Node to function in SSL mode
- Configure standalone CAST Dashboards to function in SSL mode

Which CAST applications can currently connect in SSL mode?

See the table in **CAST Storage Service - Deployment requirements**.

Configuration process

Generate SSL certificates/keys

The first step is to generate the SSL certificates/keys that are required by both the CAST Storage Service/PostgreSQL instance and CAST applications.

Install OpenSSL

To generate the required SSL certificates, **OpenSSL** must be installed on the server hosting your CAST Storage Service/PostgreSQL instance.

OpenSSL installation on Windows

- Download and install OpenSSL from **http://gnuwin32.sourceforge.net/packages/openssl.htm** (note that this will install an older release - 0.9.8).
- Create a **System** and **User** Environmental variable as follows:
  - Variable Name - **OPENSSL_CONF**

- Variable Value - location of the **openssl.cnf** file, for example: **%PROGRAMFILES%\GnuWin32\share\openssl.cnf**

OpenSSL installation on Linux

OpenSSL may be preinstalled on your chosen Linux distribution. To check, run the following command:

```
openssl version
```

If OpenSSL is not installed, follow the appropriate installation instructions. For example:

```
Debian based: apt-get install openssl
RedHat/CentOS based: yum install openssl
```

Certificate/key generation

- You may wish to consult PostgreSQL documentation for more information **https://www.postgresql.org/docs/9.6/ssl-tcp.html**.
- If you wish to generate the certificates and keys on a Microsoft Windows host, you can use the following pre-defined batch file: **CertificateGeneration.bat**. Note that this file assumes that the **GnuWin32** tool is installed on the host.

Create a root key and certificate for the server hosting CAST Storage Service/PostgreSQL (root.crt, root.key) - ensure you change any settings, in particular the -**subj option** to suit your own environment:

```
openssl genrsa -des3 -out root.key 1024
openssl rsa -in root.key -out root.key
openssl req -new -key root.key -days 365 -out root.crt -x509 -subj "/CN=root.yourdomain.com"
```

Create a server key and certificate for the server hosting CAST Storage Service/PostgreSQL (server.crt, server.key) - ensure you change any settings, in particular the -**subj option** to suit your own environment:

```
openssl genrsa -des3 -out server.key 1024
openssl rsa -in server.key -out server.key
openssl req -new -key server.key -out server.csr -subj "/CN=server.hostname"
openssl x509 -req -in server.csr -CA root.crt -CAkey root.key -days 365 -out server.crt -CAcreateserial
```

Create the client certificates/keys to be used with CAST applications (postgresql.crt, postgresql.key, postgresql.pk8, postgresql.pfx) - ensure you change any settings, in particular the -**subj option** to suit your own environment:

```
openssl genrsa -des3 -out postgresql.key 1024
openssl rsa -in postgresql.key -out postgresql.key
openssl req -new -key postgresql.key -out postgresql.csr -subj "/CN=operator"
openssl x509 -req -in postgresql.csr -CA root.crt -days 365  -CAkey root.key -out postgresql.crt -CAcreateserial
openssl pkcs8 -topk8 -in postgresql.key -out postgresql.pk8 -outform der -nocrypt
openssl pkcs12 -export -out postgresql.pfx -inkey postgresql.key -in postgresql.crt -password pass:
```

Finally, copy the following "server" certificates and keys to the folder in which your **postgresql.conf** file is located:

- **root.crt**
- **server.key**
- **server.crt**

The **postgresql.conf** file located is here:

```
Windows: %PROGRAMFILES%\CAST\CASTStorageService3\db_data

Linux: Run the following commands in psql on the host server to locate the postgresql.conf file:

psql -U postgres
show config_file;
```

Configure CAST Storage Service/PostgreSQL to accept incoming SSL connections

Edit postgresql.conf to enable SSL

Edit the **postgresql.conf** file located here:

```
Windows: %PROGRAMFILES%\CAST\CASTStorageService3\db_data

Linux: Run the following commands in psql on the host server to locate the postgresql.conf file:

psql -U postgres
show config_file;
```

Modify the file as follows and then save the file:

```
# - Security and Authentication -

ssl = on                                                # (change requires restart)
ssl_cert_file = 'server.crt'        # (change requires restart)
ssl_key_file = 'server.key'                 # (change requires restart)
ssl_ca_file = 'root.crt'                # (change requires restart)
```

Edit pg_hba.conf file to enable SSL

Edit the **pg_hba.conf** file located here:

```
Windows: %PROGRAMFILES%\CAST\CASTStorageService3\db_data

Linux: Run the following commands in psql on the host server to locate the postgresql.conf file:

psql -U postgres
show hba_file;
```

Modify the file as follows to allow IPv4 and IPv6 (where appropriate) connections using SSL by adding "**hostssl**" entries and an appropriate authentication METHOD (see **https://www.postgresql.org/docs/9.6/auth-pg-hba-conf.html** for more information about this):

```
# TYPE  DATABASE                USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local           all             all                                     peer
# IPv4 local connections:
host            all             all             127.0.0.1/32            md5
host            all             all             0.0.0.0/0               md5
# Allow IPv4 loopback with SSL + password + a check on SSL cert
hostssl                 all             all             127.0.0.1/32            md5 clientcert=1
# Allow any IPv4 with SSL + password + a check on SSL cert
hostssl                 all                                     all                                     0.0.0.0
/0              md5 clientcert=1
# IPv6 local connections:
host            all             all             ::1/128                 md5
host            all             all             ::0/0                       md5
# Allow IPv6 loopback with SSL + password + a check on SSL cert
hostssl     all             all             ::1/128                 md5 clientcert=1
# Allow any IPv6 with SSL + password + a check on SSL cert
hostssl     all             all             ::0/0                   md5 clientcert=1
```

- The **hostssl** entries given above are purely **for example only**. Please ensure that you tailor this file to your own environment and that the authentication METHOD is appropriate.
- The mix of **host** and **hostssl** entries above will allow **both SSL** and **non-SSL connections**. If you prefer to block non-ssl connections, comment out all the lines starting with "host" by adding a # at the start of the line.

Restart CAST Storage Service/PostgreSQL instance

Finally restart your CAST Storage Service or PostgreSQL instance to ensure the changes you have made are taken into account.

Configure applications to function in SSL mode

To force CAST AIP core and related applications/CAST Imaging to connect to CAST Storage Service or PostgreSQL in SSL mode, the following is required:

- an `.ini` file needs to be created and then configured on all AIP Core instances (AIP Nodes) / CAST Imaging instances
- a Windows environment variable (System or User according to your own requirements) needs to be created referencing the .ini file on all AIP Core instances (AIP Nodes) / CAST Imaging instances
- configure the AIP Node to use SSL mode

Create and configure the .ini file on all AIP Core (AIP Node) instances / CAST Imaging instances

Supported CAST applications are configured to look for an **.ini** via an **environment variable** that is defined on the host server. This .ini file allows you to configure SSL mode for multiple CAST Storage Services/PostgreSQL instances and determine where the required client side certificates and keys are located.

You can store the .ini file:

- **on the server's local file system** - in this situation, all servers hosting CAST applications that must use SSL mode to connect to a CAST Storage Service/PostgreSQL instance must have a copy of the SSLParameters.ini file, therefore if you need to make an update to the file, you will need to make the update on all servers where the file exists.
- **on a shared network drive (recommended)** - in this situation, the .ini file is stored on a shared network drive that is accessible from all servers hosting CAST applications that must use SSL mode to connect to a CAST Storage Service/PostgreSQL instance. The advantage of this is that there is only one copy of the SSLParameters.ini file and the configuration is valid for all servers.

The .ini file can use **any name** (e.g. **myfile.ini**), however, a feature to enable the encryption of analyzed source code **also uses an .ini file in exactly the same way**, therefore you may already have an .ini file available if you have enabled this - see **Storing analyzed source code in encrypted format**. If this is the case, you can re-use this file and you can **mix and match configuration** from both features in this file.

Create the .ini file using a text editor and use the following syntax:

```
[HOST:PORT,database]
ssl=true
sslmode=require
sslrootcert=root.crt
ssljdbckey=postgresql.pk8
sslkey=postgresql.key
sslcert=postgresql.crt
sslpfx=postgresql.pfx
```

[HOST:PORT,database]	Refers to the target CAST Storage Service/PostgreSQL instance. For example: • SERVER1:2282,postgres • 192.168.20.52:2282,postgres • SERVER2:5432,mydb You can add as many [HOST:PORT,database] sections as you require. Note that if you omit the **database** value, by default "**postgres** " will be assumed.
ssl=	Signifies that the connection to the target server specified in HOST:PORT,database must use SSL mode (**true**).
sslmode=	Value should be one of the following: • require • verify-ca • verify-full • disable More details about this is mentioned in **https://www.postgresql.org/docs/13/libpq-ssl.html**. This variable will match the PostgreSQL SSL parameter PGSSLMODE.

All other options	Refers to the location of the **client certificates and keys** generated previously with OpenSSL The certificates and keys need to be copied to the location you have chosen. The location of these certificates and keys is flexible: • They can be stored on a shared network drive • They can be stored on the server's local file system - in this situation, the certificates and keys need to be available on all server's that must use SSL mode You can use the following path syntax: • **Local drive:** D:\keys\postgresql.key • **Direct path:** \\SERVER\certs\root.crt • **Mapped drive on server:** S:\\certs\root.crt >>> Appropriate permissions required. CAST highly recommends that you separate the storage and place the **.crt files in one folder** and **all other keys (.pk8, .key, .pfx) in another folder** - i.e. do not mix the two together.
sslrootcert=	Give the full path of root.crt file (the crt file which is generated using OpenSSL for trusted certificate authorities). This variable will match the PostgreSQL SSL parameter PGSSLROOTCERT.
sslkey=	Give the full path of postgresql.key file (the key file which is generated using OpenSSL for client certificate). This variable will match the PostgreSQL SSL parameter PGSSLKEY.
sslcert=	Give the full path of postgresql.crt file (the key file which is generated using OpenSSL for client private key). This variable will match the PostgreSQL SSL parameter PGSSLCERT.
ssljdbckey=	Give the full path of postgresql.pk8 file (this is the key file which is generated using OpenSSL based on client certificate (postgresql.crt) and client private key (postgresql.key) using pk8 for JDBC connections).
sslpfx=	Give the full path of postgresql.pfx file (this is the key file which is generated using OpenSSL based on client certificate (postgresql.crt) and client private key (postgresql.key) using pkcs12 for .NET connections).

- Ensure that the service (or the user who launches the CAST applications programs) is dedicated.
- Do not put certificates and .key files in the same folder
- Create a dedicated folder for .key files and only give access to the dedicated CAST user
- Do not use a password on the .pfx file, otherwise you will have to enter the password each time you restart
- If you are using Console to manage the Node, and you have defined a **dedicated** CAST Storage Service/PostgreSQL instance just for the Measurement schema, you must ensure that you define an entry for this CAST Storage Service/PostgreSQL instance in the .ini file and this entry MUST be present on all Nodes that are being managed in Console.

For example:

One single host CAST Storage Service/PostgreSQL instance

```
[my_css1:2284,postgres]
ssl=truesslmode=require
sslrootcert=C:\certs\root.crt
ssljdbckey=C:\certs\postgresql.pk8
sslkey=C:\certs\postgresql.key
sslcert=C:\certs\postgresql.crt
sslpfx=C:\certs\postgresql.pfx
```

Two CAST Storage Service/PostgreSQL instances

```
[my_css1:2284,postgres]
ssl=truesslmode=require
sslrootcert=C:\certs\root.crt
ssljdbckey=C:\certs\postgresql.pk8
sslkey=C:\certs\postgresql.key
sslcert=C:\certs\postgresql.crt
sslpfx=C:\certs\postgresql.pfx

[my_css2:5432,postgres]
ssl=truesslmode=require
sslrootcert=C:\temp\certs\root.crt
ssljdbckey=C:\temp\postgresql.pk8
sslkey=C:\temp\postgresql.key
sslcert=C:\temp\postgresql.crt
sslpfx=C:\temp\postgresql.pfx
```

Two CAST Storage Service/PostgreSQL instances - one with encrypted source code feature ALSO enabled

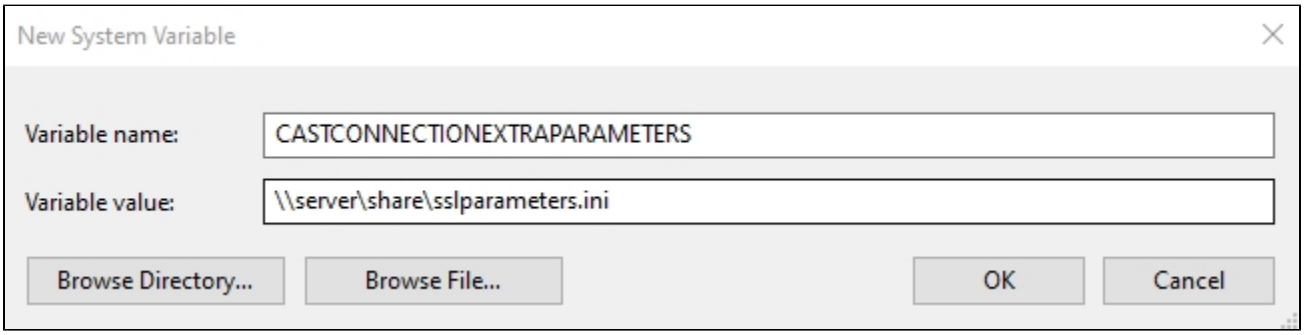
```
[my_css1:2284,postgres]
ssl=truesslmode=require
sslrootcert=C:\certs\root.crt
ssljdbckey=C:\certs\postgresql.pk8
sslkey=C:\certs\postgresql.key
sslcert=C:\certs\postgresql.crt
sslpfx=C:\certs\postgresql.pfx

[my_css2:5432,postgres]
ssl=truesslmode=require
sslrootcert=C:\temp\certs\root.crt
ssljdbckey=C:\temp\postgresql.pk8
sslkey=C:\temp\postgresql.key
sslcert=C:\temp\postgresql.crt
sslpfx=C:\temp\postgresql.pfx
encryption_key_default=AFK%3JdMEn99WypMVSCU
```

Create environment variable on all AIP Core (AIP Node) instances / CAST Imaging instances

Create a Windows environment variable (System or User according to your own requirements) on each server hosting CAST applications that must use SSL mode to connect to a CAST Storage Service/PostgreSQL instance. Use the following syntax:

- Variable Name - **CASTCONNECTIONEXTRAPARAMETERS**
- Variable Value - location of the **SSLParameters.ini** file, for example: **<FULLPATH>\SSLParameters.ini**

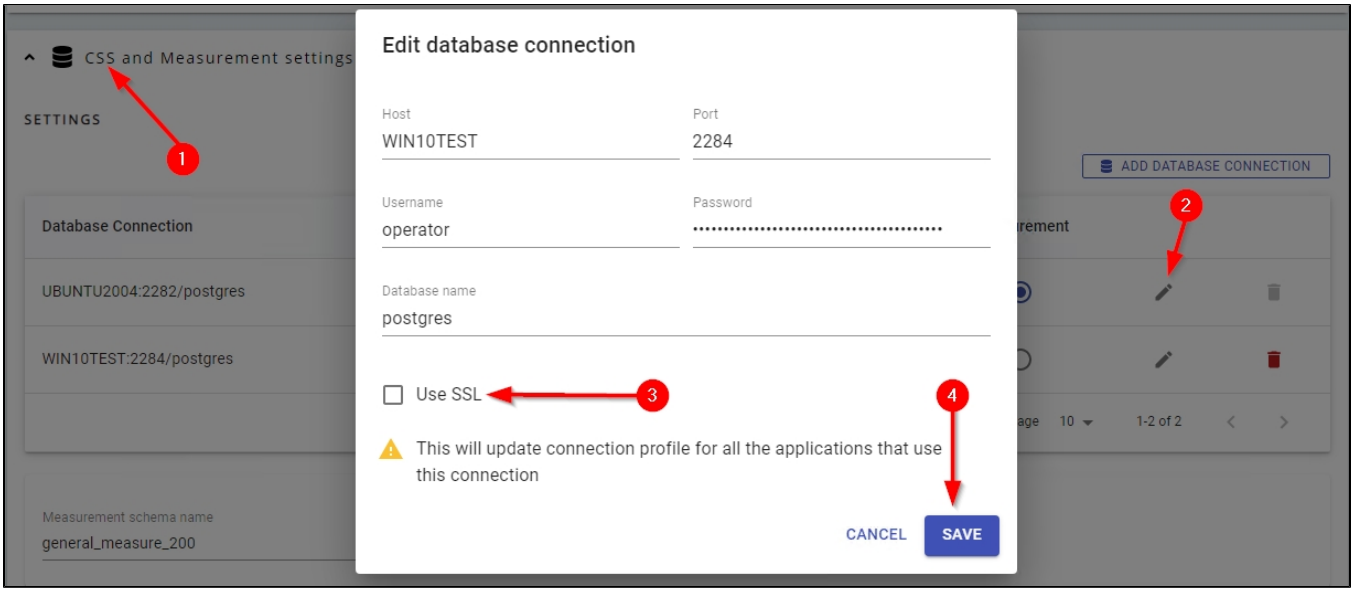


Configure AIP Node (for AIP Console) to function in SSL mode

AIP Console 2.x

Update the CAST Storage Service/PostgreSQL connection profile to use SSL mode (see **Administration Center - Settings - CSS and Measurement settings**):

Click to enlarge



AIP Console 1.x

If you are using AIP Console to manage the AIP Node, the following configuration must be performed on **each AIP Node** (i.e. a machine on which CAST AIP Core has been installed and is being managed in AIP Console) that must use an encrypted SSL connection to the target CAST Storage Service /PostgreSQL instance.

Edit the following file:

```
%PROGRAMDATA%\CAST\AipConsole\AipNode\aip-node-app.properties
```

Find the following section:

```
# ============================
# CSS Server parameters
# ----------------------------
database.server.name=my_css_server:2282
database.server.user=operator
database.server.ssl=
database.server.ssl.iniPath=
# to encrypt the password use aip-encryption-tool
database.server.password=CRYPTED2:90B1A6EC1618661401B724DB5AC34595
database.name=postgres
```

Update the **database.server.ssl** and **database.server.ssl.iniPath** parameters as follows:

```
# ============================
# CSS Server parameters
# ----------------------------
database.server.name=my_css_server:2282
database.server.user=operator
database.server.ssl=true
database.server.ssl.iniPath=<PATH_TO>/SSLParameters.ini
# to encrypt the password use aip-encryption-tool
database.server.password=CRYPTED2:90401B724DB5AC34595
database.name=postgres
```

database.server.ssl	Set this option to true to enforce an encrypted SSL connection. Without this option, a standard non-encrypted connection will be used.

database.server.ssl.iniPath	Set this to the path where your SSLParameters.ini file is located.

Restart Microsoft Windows on all AIP Nodes / CAST Imaging instances

Finally restart Microsoft Windows on all AIP Nodes/ CAST Imaging instances so that all changes are taken into account.

Configure standalone CAST Dashboards (2.x) to function in SSL mode

There are two ways to force the standalone CAST Dashboards to function in SSL mode. Both are valid for Dashboards deployed on Microsoft Windows and on Linux:

- By **modifying the URL** that points to the CAST Storage Service/PostgreSQL instance
- By **using the .ini file and an environment variable**

Method 1: Modify the connection URL

Edit the following file:

```
WAR  2.x
CATALINA_HOME\webapps\<dashboard>\WEB-INF\classes\application.properties

ZIP  2.x
<unpacked_zip>\application.properties

JAR  2.x
%PROGRAMDATA%\CAST\Dashboards\application.properties
```

Locate the following section in the file:

```
## DATASOURCE
# Resource1 is the datasource name used in domains.properties
# Adapt server name (localhost) and port (2282) if required
# You can add multiple datasources if you want to connect to multiple CSS Servers. Datasource name must be
unique
# You have to configure your domains names and relative schema names in domains.properties
restapi.datasource[0].url=jdbc:postgresql://localhost:2282/postgres
restapi.datasource[0].username=operator
restapi.datasource[0].password=CastAIP
restapi.datasource[0].poolname=Resource1
restapi.datasource[0].minimumIdle=10
restapi.datasource[0].maximumPoolSize=20
```

Modify the `restapi.datasource[0].url=jdbc:postgresql://localhost:2282/postgres` line to point to the certificate files you generated previously:

```
restapi.datasource[0].url=jdbc:postgresql://my_server:2282/postgres?
ssl=true&sslrootcert=\\\\my_host\\share\\root.crt&sslcert=\\\\my_host\\share\\postgresql.
crt&sslkey=\\\\my_host\\share\\postgresql.pk8&sslmode=verify-ca
```

This is broken down as follows:

ssl=true	Force the connection to use SSL mode.
sslrootcert=\\\\my_host\\share\\root.crt	Specifies the location of the root.crt certificate file. In this example, the file is located on a network share **\\my_host\share\root.crt**. Back slashes in the path MUST be escaped with a backslash.
sslcert=\\\\my_host\\share\\postgresql.crt	Specifies the location of the postgresql.crt certificate file. In this example, the file is located on a network share **\\my_host\share\postgresql.crt**. Back slashes in the path MUST be escaped with a backslash.
sslkey=\\\\my_host\\share\\postgresql.pk8	Specifies the location of the postgresql.pk8 certificate file. In this example, the file is located on a network share **\\my_host\share\postgresql.pk8**. Back slashes in the path MUST be escaped with a backslash.
sslmode=verify-ca	See the table in **https://www.postgresql.org/docs/13/libpq-ssl.html#LIBPQ-SSL-PROTECTION** for more information.

Finally restart the application server so that all changes are taken into account.

Method 2: Using the .ini file and an environment variable

Configure an .ini file

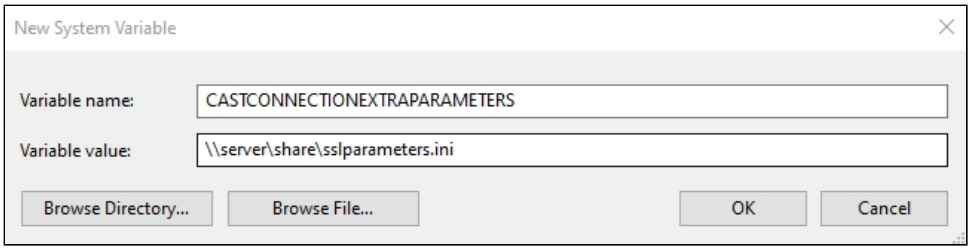
All host servers on which your CAST Dashboards are running need to have access to an **.ini file** that defines the CAST Storage Service/PostgreSQL instances which must be accessed via SSL, along with the various required certificates:

- If you stored the **.ini file** for your AIP Core instance (Node) - see **Create and configure the .ini file on all AIP Core (AIP Node) instances / CAST Imaging instances** above - on a **shared network resource** that the CAST Dashboard host server(s) has access to, you can simply re-use this .ini file by pointing the environment variable at this .ini file - CAST highly recommends this method
- Alternatively, you can create an .ini file on the **local file system of the server(s) hosting the CAST Dashboards** - this .ini file should contain a **reference to all CAST Storage Service/PostgreSQL instances** which must be accessed via SSL, along with the various required certificates - i.e. use the same syntax as described above in **Create and configure the .ini file on all AIP Core (AIP Node) instances / CAST Imaging instances** above. The disadvantage of this option is that if you need to change an encryption key (for example), you will need to update multiple .ini files.

> A feature to encrypt analyzed source code **also uses an .ini file in exactly the same way as described here**, therefore you may already have an .ini file/environment variable available if you have enabled this - see **Storing analyzed source code in encrypted format**. If this is the case, you can re-use this file and you can **mix and match configuration** from both features in this file.

Create an environment variable on the host server

On all servers hosting Dashboards that must read encrypted source code, add an environment variable called **CASTCONNECTIONEXTRAPARAMETERS** that points to the **.ini** file created previously. Use the following syntax:

Microsoft Windows	System or user environment variable: • Variable Name - **CASTCONNECTIONEXTRAPARAMETERS** • Variable Value - location of the **.ini** file, for example: **<FULLPATH>\myfile.ini** 
Linux	Set a system wide environment profile via the **/etc/environment** file: • **CASTCONNECTIONEXTRAPARAMETERS="<FULLPATH>/myfile.ini"** 

Finally restart the application server so that all changes are taken into account.

Notes

Any custom scripts that you use to connect to your CAST Storage Service / PostgreSQL instance will need to be refactored to take advantage of the encrypted SSL connection should you wish to.