



Derek Tempongko
Paolo Bruni

DB2 10 for z/OS: Configuring SSL for Secure Client-Server Communications

Introduction

This IBM® Redpaper™ publication provides information about how to set up and configure DB2® 10 for z/OS® with Secure Sockets Layer (SSL) using the z/OS V1R11 Communications Server: IP Application Transparent - Transport Layer Security (AT-TLS) services. This paper also covers the steps for configuring SSL support for non-Java DB2 clients and IBM Data Server Driver for Java Database Connectivity (JDBC) and SQLJ (Type 4 connectivity) clients to access a DB2 10 for z/OS server.

The intended audience for this paper includes network and security administrators and database administrators who want to set up and configure SSL support for DB2 10 for z/OS.

This paper discusses the following topics:

- ▶ Overview of SSL and TCP/IP AT-TLS
- ▶ Configuring DB2 10 for z/OS server with SSL support
- ▶ Configuring DB2 10 for z/OS requester with SSL support
- ▶ Configuring Java applications using IBM Data Server Driver for JDBC and SQLJ to use SSL
- ▶ Configuring non-Java DB2 clients for Linux, UNIX, and Windows to use SSL
- ▶ Configuring remote client applications to use SSL through a DB2 Connect server for Linux, UNIX, and Windows
- ▶ Client access to DB2 using digital certificates

This paper presents additional information to the more general contents of *Security Functions with DB2 10 for z/OS*, SG24-7959.

If you are at DB2 9 for the z/OS level, refer to *DB2 9 for z/OS: Configuring SSL for Secure Client-Server Communications*, REDP-4630.

Overview of SSL and TCP/IP AT-TLS

SSL is a client-server cryptographic protocol that is based on the earlier SSL specifications developed by Netscape Corporation for securing communications that use TCP/IP sockets. The SSL protocol is executed at the application level. Therefore, an application has to be designed to support SSL to benefit from the SSL protection. When z/OS V1R7 Communications Server introduced Application Transparent-Transport Layer Security (AT-TLS), which implements TLS and its predecessor SSL at the Transport Control Protocol (TCP) layer of the TCP/IP stack, DB2 was able to provide secure (encrypted) communication for remote connections (inbound and outbound) by exploiting the functions of AT-TLS. In this paper, the terms TLS and SSL are used interchangeably. The TLS/SSL service is available under z/OS and is called *System SSL*.

AT-TLS support is policy driven and managed by the z/OS Communications Server Policy Agent (PAGENT), which implements policy-based networking for the z/OS environment. For information about policy-based networking, refer to *z/OS V1R11 Communications Server IP: Configuration Guide*, SC31-8775-15.

Because AT-TLS support is policy driven, it provides application-to-application security (such as TLS) using policies. As a result, DB2 continues to send and receive clear text data over the socket while an active AT-TLS policy enables TCP/IP to invoke the services of the system SSL to protect the data being transmitted over the network. Figure 1 illustrates the flow of DB2 10 for z/OS and AT-TLS using SSL to protect the data exchanges over the network with a remote client system that also supports SSL.

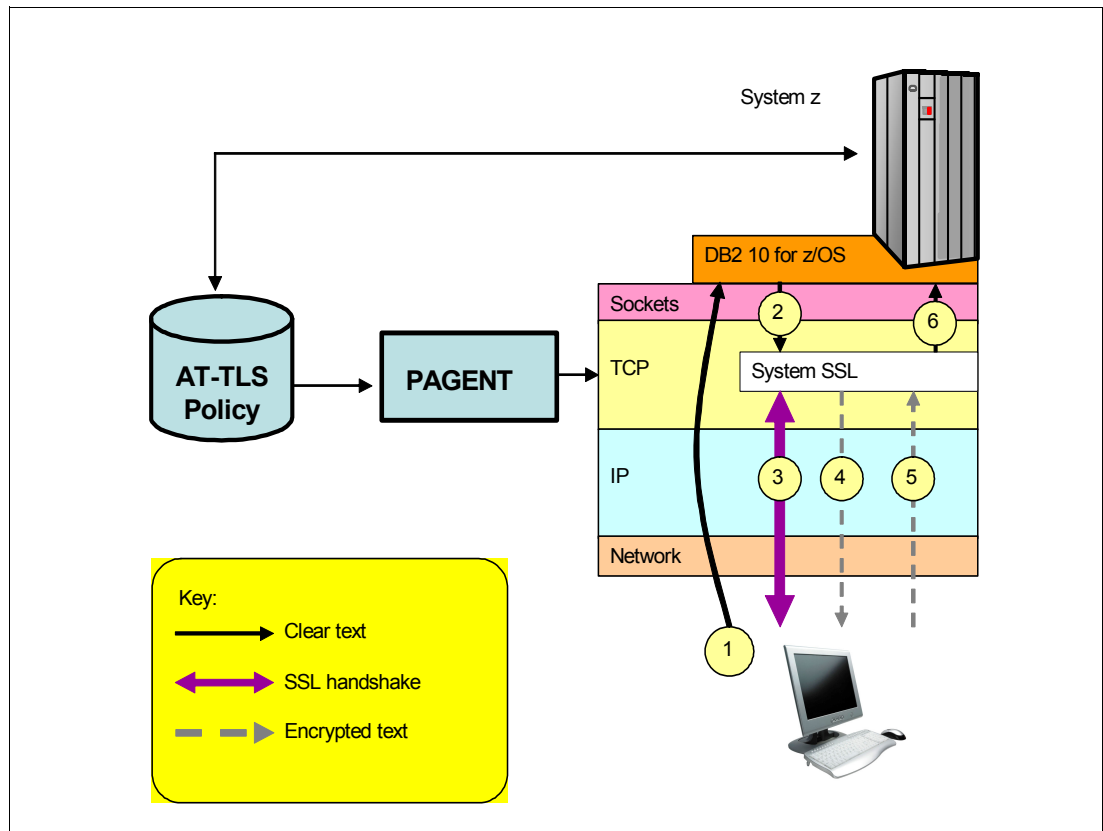


Figure 1 Flow of DB2 10 for z/OS and AT-TLS

The diagram in Figure 1 on page 2 illustrates the following flow:

1. The client connection to the DB2 server gets established in the clear (no security, TCP handshake only).
2. The DB2 server sends data in the clear and the TCP layer queues it.
3. The TCP layer invokes System SSL to perform an SSL handshake under the identity of the DB2 server, using policy information.
4. The TCP layer invokes System SSL to encrypt the queued data and sends it to the client.
5. The client sends encrypted data and the TCP layer invokes System SSL to decrypt the data.
6. The DB2 server receives data in the clear.

Configuring DB2 10 for z/OS server with SSL support

To configure DB2 10 for z/OS server with SSL support, perform the following steps:

1. Configure the policy agent as a started task in z/OS
2. Define the security authorization for PAGENT
3. Define the policy agent configuration files
4. Configure AT-TLS
5. Define the AT-TLS policy rules
6. Create digital certificates in RACF
7. Configure a secure port for the DB2 10 for z/OS server

We discuss these steps in the following sections.

Configure the policy agent as a started task in z/OS

The PAGENT runs as a UNIX process so it can be started either from the UNIX System Services shell or as a z/OS started task. In our example, we use the z/OS started task procedure to start the policy agent.

To start the policy agent as a z/OS started task, you can use the started procedure in Example 1.

Example 1 PAGENT JCL started procedure

```
//PAGENT  PROC
//PAGENT  EXEC PGM=PAGENT,REGION=0K,TIME=NOLIMIT,
//          PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//*
//* For information on the above environment variables, refer to the
//* IP CONFIGURATION GUIDE. Other environment variables can also be
//* specified via STDENV.
//*
//STDENV   DD DSN=SYS1.TCPPARMS(PAENV),DISP=SHR
//*
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively. But
//* normally, PAGENT doesn't write output to stdout or stderr.
//* Instead, output is written to the log file, which is specified
//* by the PAGENT_LOG_FILE environment variable, and defaults to
//* /tmp/pagent.log. When the -d parameter is specified, however,
```

```
/* output is also written to stdout.
/*
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//CEEDUMP DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

You can also find a sample started task procedure for PAGENT in TCPIP.SEZAINST(EZAPAGSP).

You can use environment variables that are either configured in an IBM MVS™ data set or a z/OS UNIX file that is specified by the STDENV data definition (DD) to run with the required configuration. In our example, we have configured the environment variables for PAGENT in an MVS data set, SYS1.TCPPARMS, and member PAENV, as shown in Example 2.

Example 2 SYS1.TCPPARMS(PAENV) data set containing PAGENT environment variables

```
TZ=PST8PDT7
PAGENT_CONFIG_FILE=/'SYS1.TCPPARMS(PAGENT) '
PAGENT_LOG_FILE=SYSLOGD
```

We explain the environment variables that are used in Example 2 here:

- ▶ TZ specifies the local time zone for the policy agent¹ process. Here, it is set to PST8 (Pacific Standard Time GMT-08:00) and PDT7 (Pacific Daylight Savings Time GMT-07:00).
- ▶ PAGENT_CONFIG_FILE specifies the PAGENT configuration file to use. The PAGENT configuration file is a member that is located in the data set SYS1.TCPPARMS.
- ▶ PAGENT_LOG_FILE specifies the log file name that is used by PAGENT. It is set to log policy agent messages to SYSLOGD.

Before you can start PAGENT, you need to define the appropriate security authorizations, as described in the next section.

Define the security authorization for PAGENT

The policy agent can affect system operation significantly when it is started. Therefore, we need to define a user (an authorization ID) to a security product authority, such as RACF®, to start the policy agent.

To set up the security definitions for PAGENT to RACF, perform the following steps:

1. Define the PAGENT-started task to RACF
2. Define user ID, PAGENT, for the PAGENT-started task
3. Assign user ID, PAGENT, with the PAGENT-started task
4. Permit authorized users' access to manage the PAGENT-started task
5. Restrict unauthorized users' access to UNIX pasearch command

Define the PAGENT-started task to RACF

To set up the PAGENT-started task to RACF, you need to define a profile for it to the RACF generic resource class called STARTED using the RDEFINE command.

¹ The policy agent is a component within server platforms for enforcing policy decisions regarding network resources.

SETROPTS: In Example 3, we include these SETROPTS commands for completeness. Running these commands when the STARTED class is already activated has no effect.

Example 3 shows the RACF commands used to set up the PAGENT started task.

Example 3 RACF commands to define PAGENT started task to RACF

```
//DAEMONS EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
  SETROPTS CLASSACT(STARTED)
  SETROPTS RACLIST(STARTED)
  SETROPTS GENERIC(STARTED)
  RDEFINE STARTED PAGENT.*
  SETROPTS RACLIST(STARTED) REFRESH
  SETROPTS GENERIC(STARTED) REFRESH
/*
```

If you also want to log messages through SYSLOGD, include an RDEFINE command to define a profile for SYSLOGD to the STARTED class, for example:

```
RDEFINE STARTED SYSLOGD.*
```

Define user ID, PAGENT, for the PAGENT-started task

The user ID for the PAGENT-started task must have z/OS UNIX superuser authority. That is, the z/OS UNIX UID for this user must be set to 0. UID 0 is considered a superuser. Also, you need to assign a default group (DFLTGRP) for the user ID.

Example 4 shows the RACF command that is used to define a user ID called PAGENT to a default group called OMVSGRP with an OMVS segment that specifies UID 0.

Example 4 RACF command to define a user ID for the PAGENT-started task

```
/PAUSER EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
  ADDUSER PAGENT DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/'))
/*
```

If your security administrator has defined the SHARED.IDS profile in the UNIXPRV class, the UID value must be unique. Use the SHARED keyword in addition to UID to assign a value that is already in use. Otherwise, RACF does not require the UID to be unique, so there is no need to use the SHARED keyword.

If you also are going to log messages to SYSLOGD, define a user ID with superuser authority for the SYSLOGD started task, for example:

```
ADDUSER SYSLOGD DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/'))
```

Assign user ID, PAGENT, with the PAGENT-started task

Use the RACF RALTER command to associate the user ID PAGENT that was created from the prior step to the PAGENT-started task, as illustrated in Example 5.

Example 5 RACF command to associate user ID with PAGENT-started task

```
//ASCPAUSR EXEC PGM=IKJEFT01
```

```
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
  RALTER STARTED PAGENT.* STDATA(USER(PAGENT))
/*
```

If you also are going to log messages to SYSLOGD, you need to associate the user ID SYSLOGD with the SYSLOGD started task, for example:

```
RALTER STARTED PAGENT.* STDATA(USER(SYSLOGD))
```

Permit authorized users' access to manage the PAGENT-started task

To restrict unauthorized access to manage the PAGENT-started task, we need to define a profile named MVS.SERVMMGR.PAGENT in the RACF resource class OPERCMDS and only give authorized users access to this facility. Example 6 shows the RACF commands that are used to achieve this access.

Example 6 RACF commands to permit authorized users access to manage PAGENT-started task

```
//PERMITPA EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
  SETROPTS CLASSACT(OPERCMDS)
  SETROPTS RACLIST (OPERCMDS)
  RDEFINE OPERCMDS (MVS.SERVMMGR.PAGENT) UACC(NONE)
  PERMIT MVS.SERVMMGR.PAGENT CLASS(OPERCMDS) ACCESS(CONTROL) -
    ID(PAGENT)
  SETROPTS RACLIST(OPERCMDS) REFRESH
/*
```

Restrict unauthorized users' access to UNIX psearch command

Use the UNIX **psearch** command to display policy definitions. The output from this command indicates whether policy rules are active, and the output shows the parsed results of the policy definition attributes. Note that the policy agent is designed to ignore unknown attributes, so misspelled attributes result in the default values being used. The **psearch** output can be used to verify that policies are defined correctly. However, in specific instances, not every user is permitted to read the policy definitions. To restrict unauthorized access to the **psearch** command, a profile is defined to the RACF resource class SERVAUTH. The profile name can be defined by the TCP/IP stack (TcpImage) and policy type (ptype = QoS, IDS, IPSec, or TTLS):

```
EZB.PAGENT.sysname.TcpImage.ptype
```

In this example, these variables have the following definitions:

- ▶ *sysname*: System name that is defined in the sysplex
- ▶ *TcpImage*: TCP name (TCP/IP procedure name) for the policy information that is being requested
- ▶ *ptype*: Policy type that is being requested:
 - QoS: Policy Quality of Service
 - DS: Policy IDS
 - PSec: Policy IPSec
 - TTLS: Policy AT-TLS

Tip: Wildcarding is allowed on segments of the profile name.

Example 7 shows the RACF commands that are used to restrict unauthorized access to the UNIX **pasearch** command.

Example 7 RACF commands to restrict unauthorized users access to UNIX pasearch command

```
//PAGNACC EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RDEFINE SERVAUTH EZB.PAGENT.UTEC224.TCPIP.* UACC(NONE)
PERMIT EZB.PAGENT.UTEC224.TCPIP.* CLASS(SERVAUTH) -
      ID(USRT001) ACCESS(READ)
PERMIT EZB.PAGENT.UTEC224.TCPIP.* CLASS(SERVAUTH) -
      ID(USRT002) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
/*
```

Now that we have set up all the security authorizations for the PAGENT, we need to configure the policy agent for AT-TLS.

Define the policy agent configuration files

The policy agent is responsible for reading policies from configuration files, a Lightweight Directory Access Protocol (LDAP) server, or both. However, AT-TLS policies can only be defined in configuration files. Before we can define the AT-TLS policies, we must configure certain operational characteristics of the policy agent in the main configuration file. The main configuration file can contain the following statements:

- ▶ TcplImage statement
- ▶ LogLevel statement

In the following two sections, we describe the configuration steps:

- ▶ Define the TcplImage statements
- ▶ Define the appropriate logging level

Define the TcplImage statements

The TcplImage statement specifies a TCP/IP image and its associated image-specific configuration file. Each TcplImage statement can specify the file name of an image-specific configuration file. If this file name is specified, it can specify a file name that contains image-specific configuration definitions for AT-TLS policies. If the file name is not specified, the main configuration file is also the image-specific configuration file for that stack.

CommonTTLSSConfig statement: The main configuration file optionally can contain a CommonTTLSSConfig statement that identifies a shared AT-TLS policy file for all the TCP/IP stacks in your environment.

Depending on your environment, you can define the Tcplmage statement in the following way:

- If your environment is configured with multiple TCP/IP stacks and each TCP/IP stack will have its own policy definitions, specify Tcplmage statements identifying an image-specific configuration file for each TCP/IP stack. See Figure 2 for an illustration of this type of configuration.

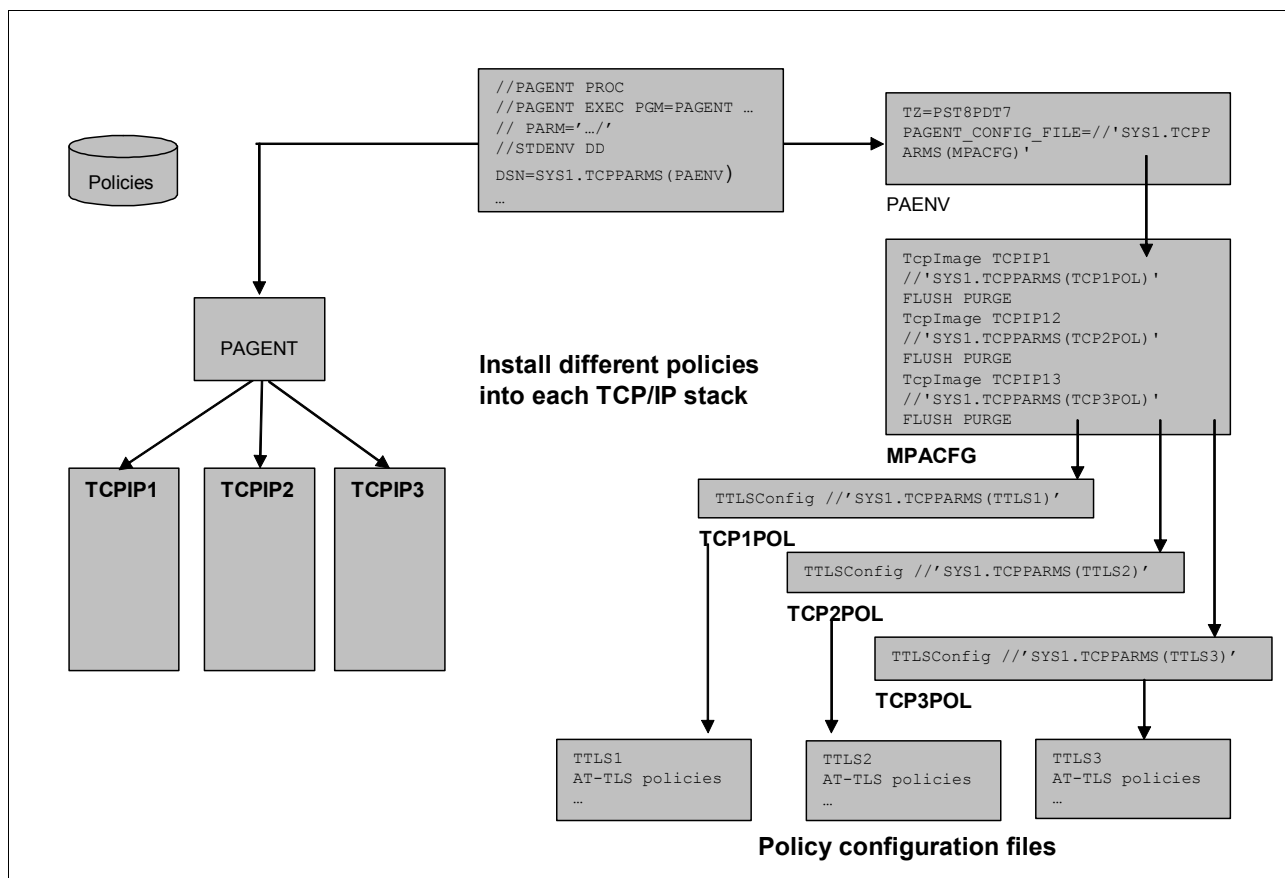


Figure 2 Multiple TCP/IP stacks and multiple AT-TLS policies

- If your environment is configured with multiple TCP/IP stacks and all the TCP/IP stacks use the same policy definitions, specify Tcplmage statements for each TCP/IP stack and omit the specification of the image-specific configuration file. The main configuration file contains a TTLSCfg statement that identifies a single policy file for all the TCP/IP stacks. See Figure 3 on page 9 for an illustration of this type of configuration.

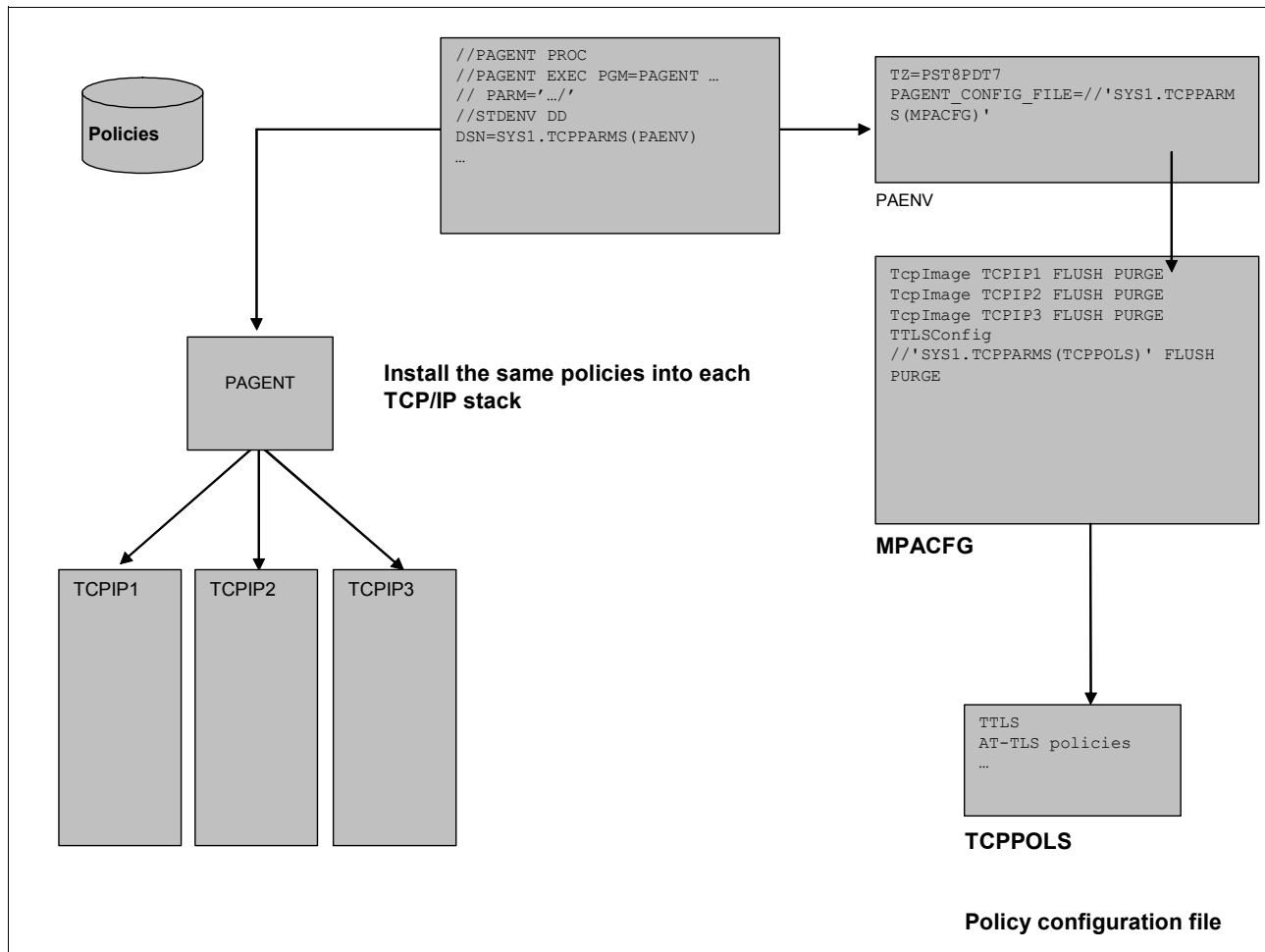


Figure 3 Multiple TCP/IP stacks and a single AT-TLS policy definition

- If your environment is configured with a single TCP/IP stack, only one AT-TLS policy definition exists. In this case, you specify the `TcpImage` statement for the single TCP/IP stack and omit the specification of the image-specific configuration file. The main configuration file contains a `TTLSCfg` statement that identifies a single policy file for the TCP/IP stack. See Figure 4 on page 10 for an illustration of this type of configuration.

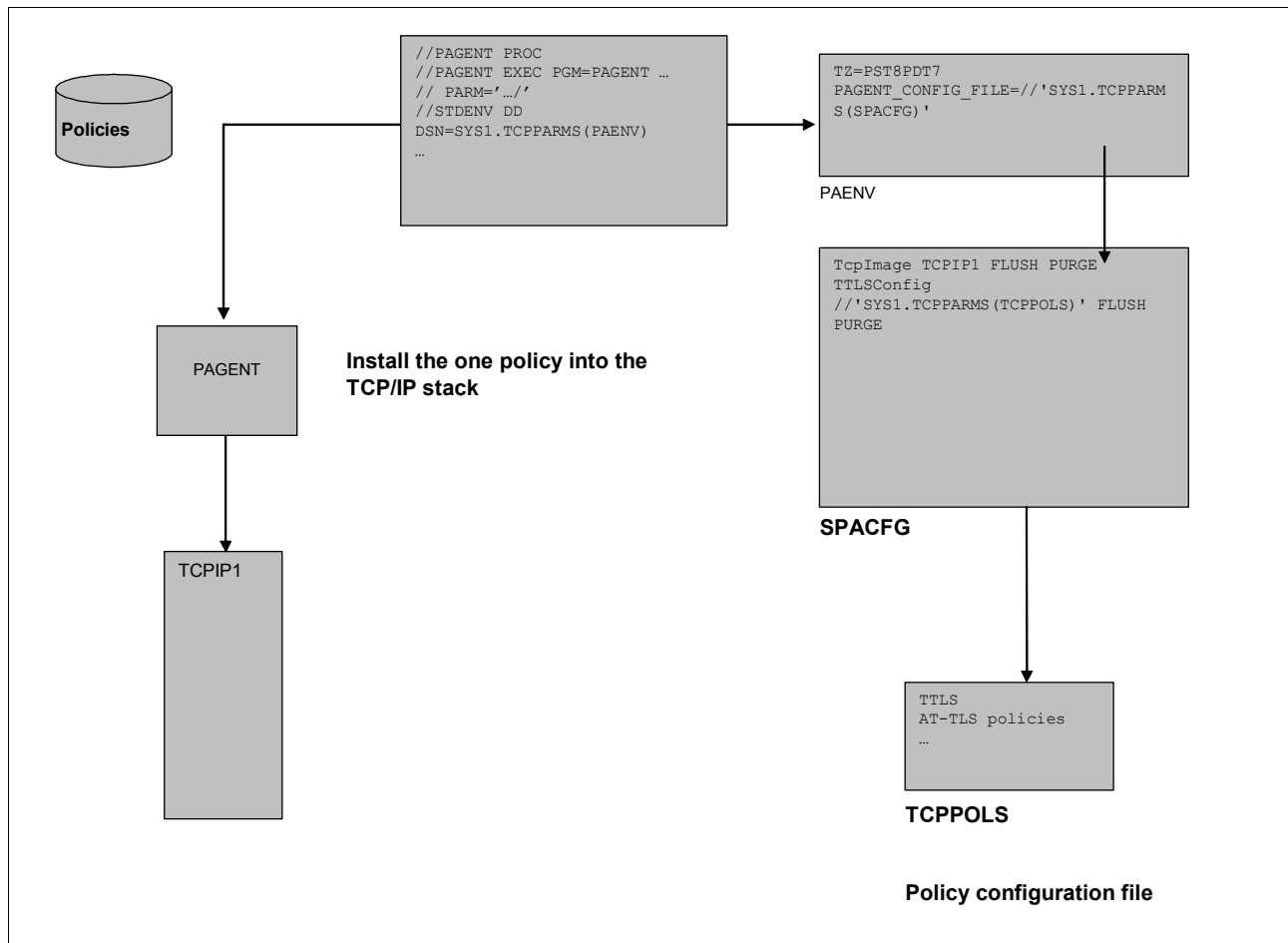


Figure 4 Single TCP/IP stack and a single AT-TLS policy

It is possible that TCP/IP stacks that are configured to the policy agent are not started or even defined. When this situation happens, the policy agent fails when trying to connect to those stacks and it logs the appropriate error messages for diagnostic purposes.

A TcpImage statement optionally can specify the parameters FLUSH/NOFLUSH or PURGE/NOPURGE. These optional parameters determine whether policies are deleted from the associated TCP/IP stack under the conditions that are detailed in Table 1 on page 11.

Table 1 How policy agent FLUSH and PURGE parameters are used

| Parameter | When used | Results |
|-------------------|---|--|
| FLUSH and NOFLUSH | Used after policies are read without errors, when triggered by the following events: <ul style="list-style-type: none"> ► Policy agent start-up ► Tcplmage statement added ► MODIFY REFRESH command issued | For FLUSH: <ul style="list-style-type: none"> ► All policies are deleted from policy agent and the TCP/IP stack ► QoS policy statistics are reset to 0 For NOFLUSH: <ul style="list-style-type: none"> ► No policies are deleted from policy agent or the TCP/IP stack Policies removed from the configuration file are not deleted from the policy agent or the TCP/IP stack |
| PURGE and NOPURGE | <ul style="list-style-type: none"> ► Policy agent shutdown ► Tcplmage statement deleted | For PURGE: <ul style="list-style-type: none"> ► All policies are deleted from the policy agent and the TCP/IP stack For NOPURGE: <ul style="list-style-type: none"> ► All policies are deleted from the policy agent ► No policies are deleted from the TCP/IP stack |

When any (or all) TCP/IP stacks are shut down, the policy agent does not shut down automatically. When the TCP/IP stacks are restarted, active policies are reinstalled automatically.

The following example defines the main configuration file that installs a common AT-TLS policy file 'SYS1.TCPPARMS(TTLSPOL)' to the TCPIP TCP/IP image, after flushing the existing policy control data when the TCPIP image is restarted:

```
TcpImage TCPIP //'SYS1.TCPPARMS(TTLSPOL)' FLUSH PURGE
```

Define the appropriate logging level

You use the LogLevel statement to define the amount of information to be logged by the policy agent. The default is to log only event, error, console, and warning messages. This level might be an appropriate level of information for a stable policy configuration, but more information might be required to understand policy processing or debug problems when first setting up policies or when making significant changes. Specify the LogLevel statement with the appropriate logging level in the main configuration file.

To define the appropriate logging level in the main configuration file, specify the LogLevel statement keyword followed by an integer that represents the level of logging to be performed by the policy agent. The following levels are supported:

| | |
|------------|---|
| 1 | SYSERR: System error messages |
| 2 | OBJERR: Object error messages |
| 4 | PROTERR: Protocol error messages |
| 8 | WARNING: Warning messages |
| 16 | EVENT: Event messages |
| 32 | ACTION: Action messages |
| 64 | INFO: Informational messages |
| 128 | ACNTING: Accounting messages |
| 256 | TRACE: Trace messages |

To specify more than one level of logging, specify the sum of all the log levels that are required in the LogLevel statement. For example, to request SYSERR messages (level 1) and EVENT messages (level 16), you specify LogLevel 17.

LogLevel 511: To request that the policy agent log everything, you can specify LogLevel 511. This level of logging can produce a significant amount of output. Consider the log size when the syslog daemon is used as the log file.

Example 8 shows the definitions of the main configuration file, PAGENT, for a single TCP/IP stack environment. The AT-TLS policies are defined in a separate image-specific configuration file, TTLSPOL, for the TCPIP TCP/IP image.

Example 8 PAGENT configuration file

```
# LogLevel statement
#   SYSERR, OBJERR, PROTERR, and WARNING messages are logged.
LogLevel 15
#
# TcpImage statement
#   TCP/IP image: TCPIP
#   Path to image-specific configuration file: SYS1.TCPPARMS(TTLSPOL)
#   FLUSH parameter specified to delete existing policy data in the
#   stack on PAGENT start-up or when the configuration files change.
#   PURGE parameter specified to delete active policy data from the
#   stack and Policy Agent when PAGENT is shut down normally.
TcpImage TCPIP //'SYS1.TCPPARMS(TTLSPOL)' FLUSH PURGE
# Here is an alternate way to specify a common AT-TLS policy for the
# TCPIP image:
#
#   TcpImage TCPIP FLUSH PURGE
#   TTLSSConfig //'SYS1.TCPPARMS(TTLSPOL)' FLUSH PURGE
#
```

We now discuss the steps to configure AT-TLS.

Configure AT-TLS

AT-TLS support is controlled by the TTLS or NOTTLS parameter on the TCPCONFIG statement in PROFILE.TCPIP. AT-TLS is enabled by specifying TCPCONFIG TTLS. The information that is required to negotiate secure connections is provided to the TCP/IP stack by AT-TLS policies that are configured in the policy agent. When AT-TLS is enabled and a newly established connection is first used, the TCP layer of the stack searches for a matching AT-TLS policy that is installed from the policy agent. If no policy is found, the connection is created without AT-TLS involvement.

Tip: To enable TTLS without modifying the PROFILE.TCPIP and stopping and starting the TCP/IP stack, define a separate file that contains the TCPCONFIG TTLS statement and issue the VARY TCPIP OBEYFILE command, as shown in this example:

OBEYFILE called TTLSON in the SYS1.TCPPARMS data set:

```
TCPCONFIG TTLS
```

On the MVS console, issue this command:

```
VARY TCPIP,,0,DSN=SYS1.TCPPARMS(TTLSON)
```

To disable TTLS, perform the following actions:

OBEYFILE called TTLSOFF in the SYS1.TCPPARMS data set:

```
TCPCONFIG NOTTLS
```

On the MVS console, issue this command:

```
VARY TCPIP,,0,DSN=SYS1.TCPPARMS(TTLSOFF)
```

Set up TTLS stack initialization access control for AT-TLS

This step is optional. When AT-TLS is started during TCP/IP stack initialization, the policy agent might not completely load all the policy information into the stack. This situation might leave a window of time where connections that need to be covered by AT-TLS remain in clear text (non-secure) connections. To control the access to the stack during this window of time, a profile is defined in RACF to the resource class SERVAUTH. The profile name can be defined by TCP/IP stack (TcpImage):

EZB.INITSTACK.sysname.TcpImage

Where the variables have these definitions:

- ▶ *sysname*: System name that is defined in the sysplex
- ▶ *TcpImage*: TCP name (TCP/IP procedure name) for policy information that is being requested

For those applications that require TCP/IP stack access before the PAGENT is started, permit READ access to the *EZB.INITSTACK.sysname.TcpImage* profile in the SERVAUTH class. Example 9 shows the RACF commands that are used to set up TCP/IP stack initialization access control for AT-TLS. This step is optional for DB2, because DB2 does not process any TCP/IP connections until the TCP/IP stack is completely initialized. At this time, all applicable AT-TLS policies have been loaded into the TCP/IP stack.

Example 9 RACF commands to set up TCP/IP stack initialization access control for AT-TLS

```
//STACKACC EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
SETROPTS CLASSACT(SERVAUTH)
SETROPTS RACLIST (SERVAUTH)
SETROPTS GENERIC (SERVAUTH)
RDEFINE SERVAUTH EZB.INITSTACK.UTEC224.TCPIP UACC(NONE)
PERMIT EZB.INITSTACK.UTEC224.TCPIP CLASS(SERVAUTH) -
ID(OMVSKERN) ACCESS(READ)
PERMIT EZB.INITSTACK.UTEC224.TCPIP CLASS(SERVAUTH) -
ID(PAGENT) ACCESS(READ)
PERMIT EZB.INITSTACK.UTEC224.TCPIP CLASS(SERVAUTH) -
```

```
ID(SYSLOGD) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
/*
```

AT-TLS configuration is complete. Next, we describe the commands to start and stop the PAGENT-started task and provide the commands to refresh policies in the PAGENT.

Starting and stopping PAGENT from z/OS

Use the MVS START command to start the policy agent as a started task, for example:

```
S PAGENT
```

To shut down PAGENT (normally), use the MVS STOP command, as shown in this example:

```
P PAGENT
```

When the policy agent shuts down normally, and if the PURGE option is specified in one of the policy configuration files, all policies are deleted from the policy agent and the TCP/IP stack.

Define the AT-TLS policy rules

An AT-TLS policy configuration file contains AT-TLS rules that define a set of conditions that are compared to connections when the policy is mapped during connect, or at the first select for readable or writable, poll for readable or writable, send, receive, or SIOCTTLSSLCTL ioctl. If a rule match is found, AT-TLS transparently provides TLS protocol control for the connection that is based on the security attributes that are specified in the actions that are associated with the rule.

An AT-TLS rule is defined with the TTLSSRule statement, which specifies a set of conditions that are compared against the connection that is being verified by TCP/IP. The set of conditions contains the following rule conditions:

| | |
|------------------------|---|
| LocalAddr | Local IP address or addresses |
| RemoteAddr | Remote IP address or addresses |
| LocalPortRange | Local port or ports |
| RemotePortRange | Remote port or ports |
| Jobname | Job name of the owning application or wildcard job name |
| Userid | User ID of the owning process or wildcard user ID |
| Direction | Inbound if applied to a passive socket (established by accept), outbound if applied to an active socket (established by connect), or both |

The TTLSSRule statement requires, at a minimum, the Direction condition and one other condition from the previous list. These considerations apply to rules:

- ▶ If a condition is not specified, that condition is not considered when comparing the rule and the connection for a match.
- ▶ Multiple values can be specified for the IP address and port conditions, either directly in the condition or as a referenced group.
- ▶ IPv6 addresses are valid in all environments.

TTLRules can be prioritized. Priority values can be integers in the range 1 to 2.000.000.000, with 2.000.000.000 being the highest priority. When assigning priorities, skip several values to allow for future rule insertion between existing rules. The policy agent orders the rules in alphabetical order within priority.

Tip: If connections can map to more than one rule, always use priority and leave priority space between rules.

When a rule match is found, policy lookup stops and the connection is assigned the associated actions within the rule. A TTLRule can reference up to three actions where each action represents a scope of control. Table 2 describes these AT-TLS actions.

Table 2 AT-TLS actions

| Action reference | Action statement | Description |
|--------------------------|----------------------|--|
| TTLGroupActionRef | N/A | This statement is required and it must specify TTLSEnabled ON or TTLSEnabled OFF. If TTLSEnabled OFF is specified, no additional action references and statements are required. If TTLSEnabled ON is specified, the AT-TLS environment action is required and the AT-TLS connection action is optional. |
| TTLSEnvironmentActionRef | N/A | This statement is only required if the AT-TLS group action specifies TTLSEnabled ON. If specified, this statement requires a key ring name (either RACF or gskkyman format), and the HandshakeRole (either Client, Server, or ServerWithClientAuth). Optionally, the AT-TLS connection action can be specified if a subset of connections must have separate parameters. |
| TTLSCONNECTIONActionRef | TTLSCONNECTIONAction | This statement is optional. |

For more information regarding these AT-TLS actions, refer to *z/OS V1R11 Communications Server: IP Configuration Guide*, SC31-8775-15.

SSL authentication levels

The Secure Sockets Layer (SSL) protocol supports server and client authentication during the handshake phase.

SSL provides server authentication as the minimum level of security. It uses the Server Authentication mechanism to secure communications between a server and its client and allows the client to validate the authenticity of the server.

SSL also provides client authentication as an additional level of authentication and access control. It enables a server to validate the certificates of a client at the server and thus prevents the client from obtaining a secure connection without an installation-approved certificate.

Client authentication is optional and, if used, can provide the following three levels of authentication:

- Level 1 authentication is performed by System SSL. A client passes a digital certificate to a server as part of the SSL handshake. To successfully pass the required authentication, the Certificate Authority (CA) that signs the client certificate must be trusted by the server. That is, the certificate for the CA must be in the key ring that the server uses and designates as trusted.

- Level 2 (addition to level 1) authentication requires that a client certificate be registered with RACF (or other System Authorization Facility (SAF)-compliant security products) and mapped to a valid user ID. When AT-TLS receives the client certificate during the SSL handshake, it queries RACF to verify that the certificate maps to a valid user ID before allowing a secure connection to be established. This level of client authentication provides additional access control at the server and ensures that the client is known to have a valid user ID on the server host.
- Level 3 (addition to levels 1 and 2) authentication provides the capability to restrict access to a server based on the user ID that is associated with a client certificate. A client can access a server only if the client itself is valid to the server, its certificate is valid, and a user ID associated with the certificate is valid. This level of authentication uses the RACF SERVAUTH general resource class to restrict access to the server based on the user ID of the client. If the SERVAUTH general resource class is not active or the SERVAUTH profile for the server is not defined, AT-TLS assumes that this level of authentication is not requested. However, if the SERVAUTH general resource class is active and the server's SERVAUTH profile is defined, a remote secure connection is established only if the user ID that is associated with the client certificate is permitted to the server's SERVAUTH profile. Otherwise, the secure connection is not established and the connection itself is dropped.

Table 3 lists the client authentication levels that AT-TLS supports.

Table 3 Client authentication levels

| Client authentication level | ClientAuthType | Client certificate | SERVAUTH class is active and server's SERVAUTH profile defined | Certificate validation |
|-----------------------------|--------------------|--------------------|--|---|
| None | PassThru | Optional | N/A | None |
| None | Full | Optional | N/A | Certificate is validated against key ring, if provided |
| Level 1 | Required (default) | Required | N/A | Certificate is validated against key ring |
| Level 2 | SAFCheck | Required | Optional | Certificate is validated against key ring and must be associated with a user ID in the security product |
| Level 3 | SAFCheck | Required | Required | Certificate is validated against key ring and must be associated to a user ID in the security product and must be permitted to access the server's SERVAUTH profile |

Coding the AT-TLS policy rules for SSL server authentication

Example 10 shows a simple AT-TLS rule definition for providing SSL/TLS data protection for a DB2 10 for z/OS server.

Example 10 Simple AT-TLS policy rule for a DB2 10 for z/OS server to support SSL connections

```

TTLSRule DB2ASecureServer
{
    LocalPortRange 448
    JobName DB2ADIST
    Direction Inbound

```



```

    TTLSGroupActionRef DB2ASecureGrpAct
    TTLSEnvironmentActionRef DB2ASecureEnvAct
}
TTLSGroupAction DB2ASecureGrpAct
{
    TTLSEnabled On
    Trace 15
}
TTLSEnvironmentAction DB2ASecureEnvAct
{
    TTLSKeyRingParms
    {
        Keyring DB2AKEYRING
    }
    HandShakeRole Server
}

```

Example 10 on page 16 specifies a TTLSRule called DB2ASecureServer. This rule specifies three conditions that are used to compare inbound connections arriving on the local port 448, which is owned by the DB2 Distributed Data Facility (DDF) address space, DB2ADIST. The TTLSGroupActionRef refers to the name, DB2ASecureGrpAct TTLSGroupAction statement. This statement enables AT-TLS and specifies a trace level of 15. This trace level enables the tracing of instances when a connection is mapped to an AT-TLS rule, when a secure connection is successfully initiated, and for major AT-TLS events. Because TTLSEnabled ON is specified in the DB2ASecureGrpAct statement, a TTLSEnvironmentAction statement, DB2ASecureEnvAct, is required. The environment action statement specifies the key ring, DB2AKEYRING, which is managed by RACF (which is discussed in “Create digital certificates in RACF” on page 20) and the HandshakeRole of server.

Coding the AT-TLS policy rules for SSL client authentication

If you need additional security, consider using client authentication. To use client authentication, you specify the HandshakeRole ServerWithClientAuth parameter for the TTLSEnvironmentAction statement in the AT-TLS policy, as shown in Example 11.

Example 11 AT-TLS HandshakeRole ServerWithClientAuth

```

TTLSEnvironmentAction DB2ServerSecureEnvAct
{
    TTLSKeyRingParms
    {
        Keyring DB2AKEYRING
    }
    HandshakeRole ServerWithClientAuth
    TTLSEnvironmentAdvancedParms
    {
        ClientAuthType SAFCheck
    }
}

```

Example 11 specifies the HandshakeRole of ServerWithClientAuth and defines a TTLSEnvironmentAdvancedParms statement to indicate the client authentication level, ClientAuthType set to the value of SAFCheck. This level of client authentication requires the client certificate to be registered with RACF and a user ID to be associated with it in RACF (discussed in “Register client CA certificate with RACF (optional)” on page 22.

Refreshing policies

After you have defined the AT-TLS rules, if PAGENT is started, you need to refresh or update the policy agent before the AT-TLS rules take effect. The MVS MODIFY command is used to refresh or update the policy agent. These commands use the following syntax:

- F PAGENT,REFRESH

The REFRESH command triggers the policy agent to reread the configuration files, and, if requested, to download objects from the LDAP server. If the FLUSH parameter was specified on the TcplImage or discipline configuration statement, the REFRESH command triggers FLUSH processing. One consequence of triggering FLUSH processing is that policy statistics being collected in the TCP/IP stack are reset, because FLUSH deletes and reinstalls all policies. Only use this command if you suspect that policies have somehow become out of sync between the TCP/IP stack and the policy agent.

- F PAGENT,UPDATE

The UPDATE command triggers the policy agent to reread configuration files and, if requested, to download objects from the LDAP server. This command differs slightly from the REFRESH command, because PAGENT only installs or removes from the stack any new, changed, or deleted policies, as appropriate. Therefore, we advise that you use this command in most cases.

Verifying policies

Use the z/OS UNIX **pasearch** command to query information from the z/OS UNIX policy agent. The command can be issued from the UNIX Systems Services shell or from the TSO/E OSHELL command. The **pasearch** output in Figure 5 on page 19 was produced by issuing the TSO/E **oshe11** command to execute the **pasearch** command specifying the **-t** option to display all AT-TLS policy entries, for example:

```
oshe11 pasearch -t
```

```

TCP/IP pasearch CS V1R9                      Image Name: TCP/IP
Date: 11/02/2009                            Time: 17:09:20
TTLS Instance Id: 1257148157
policyRule: DB2ASecureServer
Rule Type: TTLS
Version: 3                                Status: Active
Weight: 1                                ForLoadDist: False
Priority: 1                                Sequence Actions: Don't Care
No. Policy Action: 2
policyAction: DB2ASecureGrpAct
ActionType: TTLS Group
Action Sequence: 0
policyAction: DB2ASecureEnvAct
ActionType: TTLS Environment
Action Sequence: 0
Time Periods:
Day of Month Mask:
First to Last: 11111111111111111111111111111111
Last to First: 11111111111111111111111111111111
Month of Yr Mask: 1111111111
Day of Week Mask: 111111 (Sunday - Saturday)
Start Date Time: None
End Date Time: None
Fr TimeOfDay: 00:00                        To TimeOfDay: 24:00
Fr TimeOfDay UTC: 00:00                    To TimeOfDay UTC: 00:00
TimeZone: Local
TTLS Condition Summary:                    NegativeIndicator: Off
Local Address:
FromAddr: All
ToAddr: All
Remote Address:
FromAddr: All
ToAddr: All
LocalPortFrom: 448                        LocalPortTo: 448
RemotePortFrom: 0                        RemotePortTo: 0
JobName: DB2ADIST                        UserId:
ServiceDirection: Inbound
TTLS Action: DB2ASecureGrpAct
Version: 3
Status: Active
Scope: Group
TTLSEnabled: On
CtraceClearText: Off
Trace: 15
TTLSGroupAdvancedParms:
SecondaryMap: Off
SyslogFacility: Daemon
TTLS Action: DB2ASecureEnvAct
Version: 3
Status: Active
Scope: Environment
HandshakeRole: Server
TTLSKeyringParms:
Keyring: DB2AKEYRING
TTLSEnvironmentAdvancedParms:
SSLv2: Off
SSLv3: On
TLSv1: On
ApplicationControlled: Off
HandshakeTimeout: 10
ClientAuthType: Required
ResetCipherTimer: 0

```

Figure 5 Display AT-TLS policies using pasearch -t command

Common System SSL return codes

When a secure SSL connection is being established between a client and a server and the z/OS System SSL return code, 402, is encountered, this usually indicates a cipher suite could not be negotiated between the client and the server during the cipher suite exchange phase in the SSL handshake. A common cause for this to occur is the client is trying to negotiate a cipher suite which is defined as exportable. In this situation, verify that the proper FMIDs to support the encryption level are installed. Refer to *z/OS V1R11 System SSL Programming, Software Dependencies*, SC24-5901-08 for details of the FMID number.

The next step is to create the digital certificates and the key ring to store the digital certificates in RACF.

Create digital certificates in RACF

The z/OS Security Server, RACF, supports the management of digital certificates in a z/OS environment. Other security products, such as gskkyman, can also support the management of digital certificates. In this paper, we use RACF as the security product to manage digital certificates.

You can use RACF to create, register, store, and administer digital certificates and their associated private keys. You can use RACF to build certificate requests that can be sent to a certificate authority for signing. You can also use RACF to manage the key rings of stored digital certificates. Digital certificates and key rings are managed in RACF primarily by using the RACDCERT command. In this section, we describe how to use the RACDCERT command to administer digital certificates and key rings.

Using the RACDCERT command to administer digital certificates and key rings

The RACDCERT command is your primary administrative tool for managing digital certificates and the key rings of stored digital certificates. Authority to use the RACDCERT command is controlled through resources that are defined in the FACILITY class. The RACDCERT command is used to manage resources in the DIGTCERT, DIGTRING, DIGTNMAP, and USER classes.

Tip: You do not have to activate the DIGTCERT and DIGTRING classes to use the resources in these classes. However, performance is improved when you activate and RACLIST these classes.

To control the use of the RACDCERT command, authority to the IRR.DIGTCERT.function resource in the FACILITY class needs to be given to a user, unless the user has the SPECIAL attribute. One of the following authorities must be given to the user:

- ▶ READ access to the IRR.DIGTCERT.function to issue the RACDCERT command for themselves.
- ▶ UPDATE access to the IRR.DIGTCERT.function to issue the RACDCERT command for others.
- ▶ CONTROL access to the IRR.DIGTCERT.function to issue the RACDCERT command for SITE and CERTAUTH certificates. (This authority also has other uses.)

In “Coding the AT-TLS policy rules for SSL server authentication” on page 16, where we defined the AT-TLS rule DB2ASecureServer, one of the conditions specified was the JobName condition. The JobName that was specified was DB2ADIST (started task job name of the DB2 distributed data facility (DDF) address space), which indicates to AT-TLS that the

DB2ADIST started task is the owning application for this AT-TLS rule. Specifying the job name also implies that the user ID that is associated with DB2ADIST is used implicitly by AT-TLS to invoke any necessary RACDCERT commands. As a result, we need to give sufficient authority to the user ID that is associated with the DB2ADIST started task to access the IRR.DIGTCERT.function resource in the FACILITY class. The user ID associated to the started task DB2ADIST is SYSDSP. Example 12 on page 21 illustrates how to give sufficient authority to this user ID.

Example 12 Associating user ID to started task

```
//RACDCERT EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSUADS DD DSN=SYS1.UADS,DISP=SHR
//SYSLBC DD DSN=SYS1.BROADCAST,DISP=SHR
//SYSTSIN DD *
  SETROPTS CLASSACT(DIGTCERT DIGTRING)
  RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
  RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
  PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(SYSDSP) ACCESS(CONTROL)
  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(SYSDSP) ACCESS(READ)
  SETR RACLIST (DIGTRING) REFRESH
  SETR RACLIST (DIGTCERT) REFRESH
  SETR RACLIST (FACILITY) REFRESH
/*
```

Coding the RACDCERT commands to create the digital certificates and key ring

We are now ready to generate the digital certificates that are used for establishing the secure connection with DB2 at the server. We provide examples of RACDCERT commands to illustrate how to achieve this task.

To generate a certificate authority (CA) certificate in RACF, use the RACDCERT command that is shown in Example 13.

Example 13 Generate a certificate authority (CA) certificate

```
RACDCERT CERTAUTH -
  GENCERT -
  SUBJECTSDN(OU('SVL224 Server CA') -
    O('IBM') -
    L('SVL') -
    SP('CA') -
    C('USA')) -
  NOTAFTER(DATE(2030-12-31)) -
  WITHLABEL('SVL224 Server CA') -
  KEYUSAGE(CERTSIGN)
```

To generate a site certificate that is signed by the CA certificate that we created in RACF, use the RACDCERT command that is shown in Example 14.

Example 14 Generate a site certificate

```
RACDCERT ID(SYSDSP) -
  GENCERT -
  SUBJECTSDN(CN('v25ec099.svl.ibm.com')) -
```

```

        OU('UTEC224') -
        O('SVL224') -
        C('USA')) -
NOTAFTER(DATE(2030-12-31)) -
WITHLABEL('SVL224 Server Certificate') -
SIGNWITH(CERTAUTH LABEL('SVL224 Server CA'))

```

Now that we have generated the site certificate and the CA certificate, we need to create a key ring to store these digital certificates. For our example, the name of the key ring must match the value of the **Keyring** parameter that was specified in the DB2AEnvironmentAct statement within the DB2ASecureServer AT-TLS rule. Example 15 lists the RACDCERT commands to create the DB2AKEYRING and store the digital certificates.

Example 15 Create the DB2AKEYRING and store the digital certificate

```

RACDCERT ID(SYSDSP) ADDRING(DB2AKEYRING)
RACDCERT ID(SYSDSP) CONNECT(CERTAUTH LABEL('SVL224 Server CA') -
                             RING(DB2AKEYRING))
RACDCERT ID(SYSDSP) CONNECT(ID(SYSDSP) -
                             LABEL('SVL224 Server Certificate') -
                             RING(DB2AKEYRING) -
                             DEFAULT)

```

Important: The key ring name and the label names are case sensitive.

The final step is to export the CA certificate and save it in an MVS data set. Remote clients, who want to establish a secure connection with DB2, need to add (or import) this CA certificate to their key ring. Example 16 shows the RACDCERT command to export the CA certificate.

Example 16 Exporting the CA certificate

```

RACDCERT CERTAUTH -
        EXPORT(LABEL('SVL224 Server CA')) -
        DSN('USRT001.SVL224.CACERT')

```

Register client CA certificate with RACF (optional)

If SSL client authentication is configured (HandshakeRole is ServerWithClientAuth and ClientAuthType is SAFCheck), we need to perform these additional RACDCERT commands to register the client CA certificate in RACF and associate it with a user ID.

To register a client CA certificate to RACF as trusted and an associate user ID, USRT001, with the certificate, we use the RACDCERT ADD command, as shown in Example 17.

Example 17 Register a client CA certificate to RACF as trusted

```

RACDCERT ID(USRT001) ADD('USRT001.CLIENT.CA') TRUST

```

This command registers the client CA certificate in the data set 'USRT001.CLIENT.CA' to the RACF database. The certificate is owned by the RACF-defined user USRT001. The client CA certificate is also marked as trusted so that RACF can use it to verify the client certificate when it is presented to the system.

Next, we want to add this client CA certificate to our key ring, DB2AKEYRING. Example 18 shows us how to add this client CA certificate to our key ring by issuing the RACDCERT CONNECT command.

Example 18 Connecting a client CA certificate to a key ring

```
RACDCERT ID(SYSDSP) CONNECT(ID(USRT001) LABEL('LABEL00000001') -  
RING(DB2AKEYRING) USAGE(PERSONAL))
```

This step adds the client CA certificate to the server key ring DB2AKEYRING. The certificate is owned by the user USRT001.

We have completed the necessary configuration steps to create the digital certificates and the key ring to store the digital certificates in RACF.

For additional information, refer to *z/OS V1R11.0 Security Server RACF Command Language Reference*, SA22-7687-14.

Creating and activating client certificate name filters

A *certificate name filter* enables you to associate many client certificates with one user ID based on the unique user information in the certificate, such as the user's affiliation. You can create one or more certificate name filters to map a large number of client certificates to a limited number of user IDs, which helps you reduce administrative costs.

Follow these steps to create and activate a certificate name filter:

1. Create a certificate name filter by issuing the RACDCERT MAP command, as shown in Example 19.

Example 19 RACDCERT MAP command to create a certificate name filter

```
RACDCERT MAP ID(USRT001) -  
SDNFILTER('O=IBM.L=San Jose.SP=CA.C=US') -  
WITHLABEL('IBMers') TRUST
```

This command creates a new certificate name filter that is based on the subject's distinguished name in the certificate. The filter associates user ID USRT001 to any user presenting a certificate with subject name 'O=IBM.L=San Jose. SP=CA.C=US'.

2. Activate the SETROPTS RACLIST processing for the DIGTNMAP class. Using the RACDCERT MAP command to create a certificate name filter automatically generates a mapping profile in the DIGTNMAP class that represents the new filter. Both the DIGTNMAP class and the SETROPTS RACLIST processing for the DIGTNMAP class must be active before you can complete the creation of the new certificate name filter. Issue the following command to activate the SETROPTS RACLIST processing for the DIGTNMAP class:

```
SETROPTS CLASSACT(DIGTNMAP) RACLIST(DIGTNMAP)
```

3. Refresh the DIGTNMAP class.

When SETROPTS RACLIST processing for the DIGTNMAP class is active, you must refresh the DIGTNMAP class for the certificate name filter to take effect. Issue the following command to refresh the DIGTNMAP class:

```
SETROPTS RACLIST(DIGTNMAP) REFRESH
```

4. Register a client CA certificate to use with the certificate name filter.

During the SSL handshake phase of establishing a secure connection, AT-TLS retrieves certificate information from RACF if client authentication is specified. In order for AT-TLS to

retrieve the client CA certificate and private keys from RACF, the client CA certificate must be connected to the server key ring. You can use the new certificate name filter to register the client CA certificate to RACF, connect to the server key ring, and map to the user ID CERTAUTH as trusted by issuing the following command:

```
RACDCERT CERTAUTH ADD('USRT001.CLIENT.CRT') TRUST
RACDCERT ID(SYSDSP) CONNECT(CERTAUTH LABEL('LABEL00000001'))
RING(DB2SERVERKEYRING) USAGE(CERTAUTH))
```

This command registers the client CA certificate to RACF and maps it to the ID CERTAUTH as TRUST. It then adds the certificate to key ring DB2AKEYRING, which is owned by ID SYSDSP, and indicates that it is used for certificate authority purposes. As a result, when a remote client establishes a secure connection with DB2, AT-TLS is able to authenticate the client from the client CA certificate in RACF. Because the certificate name filter is active, user ID USRT001 is returned by AT-TLS to DB2.

Configure a secure port for the DB2 10 for z/OS server

With DB2 10 for z/OS, DDF is capable of (optionally) listening to a secondary secure port for inbound SSL connections. The DDF SQL TCP/IP Listener accepts regular (non-SSL) connections on the Distributed Relational Database Architecture (DRDA®) port, whereas the secure port accepts only SSL connections to provide secure communications with a partner that uses the SSL protocol. In other words, DB2 cannot accept a non-secure TCP/IP connection to access DB2 through the secure port. If an improper use of a secure port is detected, DB2 rejects the connection with a DSNL030I message with reason code 00D31205. Hence, client connections are assured of getting the SSL protocol connection that they require when they connect to a DB2 10 for z/OS server through a secure port. The system administrator specifies the secure port number.

To define a secure port to DB2, you can specify the secure port number during the DB2 installation by using the Distributed Data Facility Panel 2 (DSNTIP5). Or, you can specify the secure port by updating the DDF communication record in the bootstrap data set (BSDS) by using the Change Log Inventory (DSNJU003) utility.

Specifying the secure port during DB2 installation

To specify the secure port during DB2 installation, specify the TCP/IP port number in the DRDA SECURE PORT field of the Distributed Data Facility Panel 2 (DSNTIP5). For example, if you want to configure the secure port number as 448, you specify the value 448 in the SECURE PORT field, as shown in Figure 6.


```

DSNTIP5          INSTALL DB2 - DISTRIBUTED DATA FACILITY PANEL 2
===> _

Enter data below:

 1 DRDA PORT          ===> 446      TCP/IP port number for DRDA clients.
                                     1-65534 (446 is reserved for DRDA)
 2 SECURE PORT        ===> 448     TCP/IP port number for secure DRDA
                                     clients. 1-65534 (448 is reserved
                                     for DRDA using SSL)
 3 RESYNC PORT        ===> 5001     TCP/IP port for 2-phase commit. 1-65534
 4 TCP/IP ALREADY VERIFIED ===> NO  Accept requests containing only a
                                     userid (no password)? YES or NO
 5 EXTRA BLOCKS REQ   ===> 100     Maximum extra query blocks when DB2 acts
                                     as a requester. 0-100
 6 EXTRA BLOCKS SRV   ===> 100     Maximum extra query blocks when DB2 acts
                                     as a server. 0-100
 7 AUTH AT HOP SITE    ===> BOTH    Authorization at hop site. BOTH or RUNNER.
 8 TCP/IP KEEPALIVE    ===> 120     ENABLE, DISABLE, or 1-65534
 9 POOL THREAD TIMEOUT ===> 120     0-9999 seconds

PRESS:  ENTER to continue  RETURN to exit  HELP for more information

```

Figure 6 DSNTIP5: Install DB2 - Distributed Data Facility Panel 2

If the DRDA SECURE PORT field is left blank, SSL verification support is disabled and the DDF TCP/IP SQL Listener does not accept any inbound SSL connections on the secure port.

DRDA port and SSL: If SSL verification support is disabled (that is, SECPORT=0), the client can still use the DRDA PORT, and use SSL on it, but DB2 does not validate whether the connection uses SSL protocol.

Specifying the secure port by updating the DDF communication record in the BSDS

To specify the secure port by updating the DDF communication record in the BSDS, use the SECPORT parameter in the DDF statement that is used with the Change Log Inventory (DSNJU003) stand-alone utility. The SECPORT parameter specifies the port number for the DDF TCP/IP SQL Listener to accept inbound SSL connections. For example, to configure the secure port with a value of 448, you code the following line:

```
DDF LOCATION=LOC1,SECPORT=448
```

If the value of the SECPORT secure port is the same as the value of PORT or RESPORT, DB2 issues an error. If you specify a value of 0 for the SECPORT parameter, SSL verification support is disabled and the DDF TCP/IP SQL Listener does not accept any inbound SSL connections on the secure port.

DRDA port and SSL: If SSL verification support is disabled (that is, SECPORT=0), the client can still use the DRDA PORT, and use SSL on it, but DB2 does not validate whether the connection uses SSL protocol.

Data sharing considerations

For a data sharing environment, each DB2 member with SSL support must specify a secure port. The secure port for each DB2 member of the group needs to be the same, just as the DRDA PORT for each member also needs to be the same. If each DB2 member specifies a unique secure port, unpredictable behaviors might occur. For instance, sysplex member workload balancing might not function correctly.

Similarly, for DB2 members that are defined as a subset of the data sharing group, each DB2 member that belongs to the subset needs to configure the alias secure port. Use the DSNJU003 stand-alone utility and specify the value of the alias secure port in the ALIAS keyword of the DDF statement. For example, you might specify the following DDF statement to define an alias secure port of 6448 for a DB2 member that belongs to a subset:

```
DDF LOCATION=LOC1,SECPORT=448,ALIAS=LOC1ALIAS1:6446:6448
```

Tip: If you only want to define a location alias name for a DB2 member, you do not need to define a separate unique secure port for the location alias.

DB2 10 for z/OS provides an alternate method to define an alias secure port by using the -MODIFY DDF command. For example, to specify the secure port dynamically for a defined alias for a DB2 member that belongs to a subset, you can issue the following DB2 command:

```
-MODIFY DDF ALIAS(LOC1ALIAS1) SECPORT(6448)
```

If the alias is not ready to accept connections, issue the following DB2 command:

```
-MODIFY DDF ALIAS(LOC1ALIAS1) START
```

In a data sharing environment, your PROFILE.TCPIP contains PORT statements for xxxxDIST jobnames. If these PORT statements also specify the BIND ipaddr option, you need to remove the BIND ipaddr specification from the PORT statement to configure DB2 to accept secure connections through the secure port. The DDF SQL TCP/IP Listener only listens to SSL inbound socket requests on an IP address that is bound to INADDR_ANY (IPv4) or in6addr_any (IPv6). If DDF detects that an IP address, other than INADDR_ANY or in6addr_any, is bound to the secure port, DDF prevents the DDF SQL TCP/IP Listener from accepting secure connections through the secure port. The message DSNL512I condition and the error message "BINDSPECIFIC NOT SUPPORTED WITH SECURE PORT" are issued on the system console.

If you need to configure specific IP addresses for each DB2 member (for example, dynamic virtual IP address (DVIPA) configuration), you can use the BSDS DDF communication record statements IPV4/GRPIPV4 or IPV6/GRPIPV6 to define the specific IP addresses for each member. You can use the group IP address with the IP addresses that are specified for the PORT statement from the TCP/IP profile (PROFILE.TCPIP). Next, you remove the specification of the IP addresses from the PORT statement in the PROFILE.TCPIP.

Configuring DB2 10 for z/OS requester with SSL support

A DB2 requester must be able to specify an SSL-protected connection to certain servers. To ensure SSL-protected connections, you can make communications database (CDB) changes that indicate that SSL-protected connections are required to certain remote locations. If a secure connection is required, DDF must determine whether an AT-TLS policy rule has been defined and AT-TLS has been enabled for the outbound connection. DDF validates the secure status of the outbound connection by querying AT-TLS. Therefore, if AT-TLS indicates that the status of the connection is not secure, and a secure connection is required, DDF does not

establish the outbound connection with the target server. SQLCODE -904, with reason code 00D31205, is returned back to the application.

Changes in the DB2 communication database

To configure an outbound connection that requires a secure connection, insert a row in the SYSIBM.LOCATIONS table and specify the value of Y in the SECURE column. You also need to specify a port number in the PORT column for the outbound connection to use. For secure connections, the value of the PORT column must take the value of the configured secure port at the target server. However, if the value of the PORT column is blank and the value of the SECURE column specifies Y, DDF uses the reserved secure port (448) as the default.

LOCATION ALIAS name considerations for SSL support

Certain DB2 applications might require SSL protection and accept the performance cost for this level of security. However, applications might be satisfied with unprotected connections. You can achieve this flexibility by using the LOCATION ALIAS name feature.

Consider a DB2 server that is configured to support both non-secure and secure connections. At the DB2 requester, you can define two rows in the SYSIBM.LOCATIONS table: one row that specifies the location name and the non-secure DRDA port of the server and one row that specifies a separate location name and the secure DRDA port of the server and SECURE='Y'. At the DB2 server, you can define a LOCATION ALIAS name to provide an alternate name for any DB2 requesters that need to access the server using the SSL protocol. See Figure 7 on page 27.

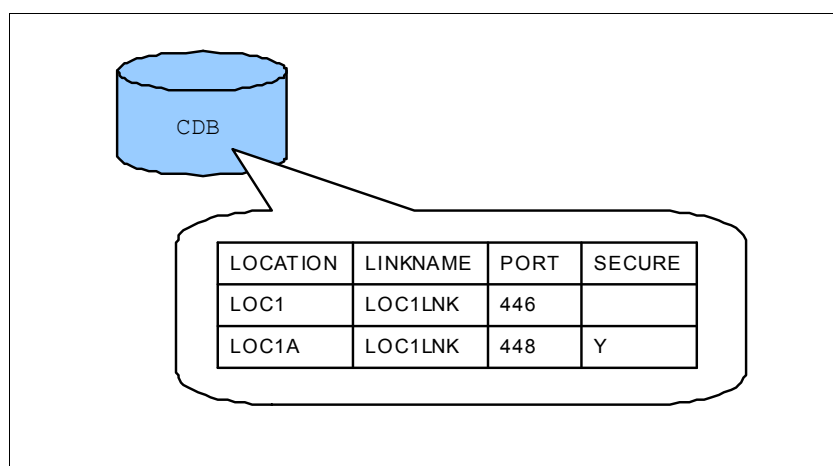


Figure 7 LOCATION ALIAS consideration

DBALIAS name considerations for SSL support

Using the LOCATION ALIAS name feature as described requires server changes. If your system environment does not allow you to perform the server changes (to define an alias name for the server), an alternate solution to the use of the LOCATION ALIAS name feature is to use the DBALIAS name feature. Therefore, using the same scenario, at the DB2 requester, you can define two rows in the SYSIBM.LOCATIONS table, as shown in Figure 8:

- ▶ A row that specifies the location name and the non-secure DRDA port of the remote server
- ▶ Another row that specifies a separate location name and the secure DRDA port and SECURE='Y' for the remote server, and a database alias (DBALIAS) name that refers to the location name from the previous row

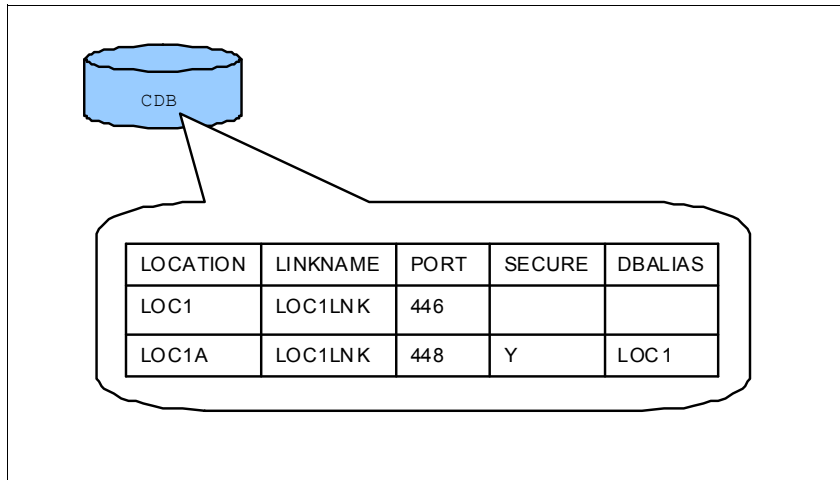


Figure 8 LOCATION DBALIAS name feature

Configure the policy agent and AT-TLS

At the DB2 requester, the z/OS system also needs to configure the policy agent and AT-TLS. The configuration steps are the same steps for configuring the server (as described in the previous sections). Follow the steps from those sections to configure the policy agent and AT-TLS.

Define AT-TLS policy rules

The steps to define the AT-TLS policy rules are the same as the steps that were described in the previous section for defining AT-TLS policy rules at the server, with the exception of these TTLSRule conditions:

- ▶ RemotePortRange = 448
- ▶ Direction = Outbound
- ▶ HandshakeRole = Client

Example 20 provides a simple AT-TLS policy rule that provides secure outbound connections to remote servers listening on port 448. The key ring name is DB2BKEYRING.

Example 20 AT-TLS policy rules for DB2 requester

```
TTLSRule DB2BSecureServer
{
  RemotePortRange 448
  JobName DB2BDIST
  Direction Outbound
  TTLSGroupActionRef DB2BSecureGrpAct
  TTLSEnvironmentActionRef DB2BSecureEnvAct
}
TTLSGroupAction DB2BSecureGrpAct
{
  TTLSEnabled On
  Trace 15
}
TTLSEnvironmentAction DB2BSecureEnvAct
{
  TTLSKeyRingParms
  {
```

```

        Keyring DB2BKEYRING
    }
    HandShakeRole Client
}

```

Digital certificate and key ring considerations

When the server is configured with HandshakeRole Server and the client system is HandshakeRole Client, the client system must have a key ring that contains the root certificates that are required to authenticate the server's certificate. The client does not need access to the private keys of any certificate.

To obtain the root certificates that are required to authenticate the server's certificate at the client, use FTP to get the server's certificate. In Example 16 on page 22, we exported the server's CA certificate, which is saved in the MVS data set USRT001.SVL224.CACERT. So, at the client, use the TSO FTP command to GET this file. The FTP mode of transmission must be in ASCII. Do not specify BINARY mode.

We use the RACDCERT command to create the key ring and store the server's CA certificate in RACF and add the server's CA certificate into the key ring. Example 21 shows the RACDCERT commands to perform these tasks.

Example 21 Create the key ring and store and add the server's CA certificate

```

RACDCERT ID(SYSDSP) ADDRING(DB2BKEYRING)
RACDCERT ID(SYSDSP)ADD('USRT001.SVL224.CACERT') TRUST -
    WITHLABEL('SVL224 Server CA')
RACDCERT ID(SYSDSP)CONNECT(ID(SYSDSP) -
    LABEL('SVL224 Server CA') -
    RING(DB2BKEYRING))

```

The final step is to update the policy agent at the client system using the MVS MODIFY command, or use the MVS START command if the policy agent is not started.

When the server is configured with HandshakeRole ServerWithClientAuth, it might need additional root certificates on its key ring to authenticate client certificates. The server decides whether a client must present a user certificate and what constitutes an acceptable certificate.

The partner application that is designated as the HandshakeRole Client decides whether to present a user certificate when challenged. To present a user certificate, it must have a private key ring with that user certificate. It must have access to the private keys of the user certificate. The client key ring must also contain the required root certificates to authenticate the server's certificate. To create the user and CA certificate at the client system, consider the RACDCERT commands that are listed in Example 22.

Example 22 Create the user and CA certificate at the client system

```

RACDCERT CERTAUTH -
    GENCERT -
    SUBJECTSDN(OU('SVL315 Client CA') -
        O('IBM') -
        L('SVL') -
        SP('CA') -
        C('USA')) -
    NOTAFTER(DATE(2030-12-31)) -
    WITHLABEL('SVL315 Client CA') -

```

```

        KEYUSAGE(CERTSIGN)
RACDCERT ID(SYSDSP) -
        GENCERT -
        SUBJECTSDN(CN('utec315.svldev.svl.ibm.com') -
                OU('UTEC315') -
                O('SVL315') -
                C('USA')) -
        NOTAFTER(DATE(2030-12-31)) -
        WITHLABEL('SVL315 Client Certificate') -
        SIGNWITH(CERTAUTH LABEL('SVL315 Client CA'))
RACDCERT ID(SYSDSP) -
        CONNECT(CERTAUTH -
                LABEL('SVL315 Client CA') RING(DB2BKEYRING))
RACDCERT ID(SYSDSP) CONNECT(ID(SYSDSP) -
        LABEL('SVL315 Client Certificate') -
        RING(DB2BKEYRING) DEFAULT)
RACDCERT CERTAUTH EXPORT(LABEL('SVL224 Client CA')) -
        DSN('USRT002.SVL315.CACERT')

```

At the server, store the client CA certificate to RACF and add it to the server key ring (DB2AKEYRING) with the RACDCERT commands that are listed in Example 23 on page 30.

Example 23 Store the client CA certificate and add it to the server key ring

```

RACDCERT ID(SYSDSP)ADD('USRT002.SVL315.CACERT') TRUST -
        WITHLABEL('SVL315 Client CA')
RACDCERT ID(SYSDSP)CONNECT(ID(SYSDSP) -
        LABEL('SVL315 Client CA') -
        RING(DB2AKEYRING))

```

Configuring Java applications using IBM Data Server Driver for JDBC and SQLJ to use SSL

In this section, we describe the steps to configure a Java application that uses IBM Data Server Driver for JDBC and SQLJ (Type 4 connectivity) to access a DB2 10 for z/OS (or later versions) server using SSL. Two configuration steps are necessary:

- ▶ Configure connections under the IBM Data Server Driver for JDBC and SQLJ to use SSL
- ▶ Configure the Java Runtime Environment to use SSL

Configure connections under the IBM Data Server Driver for JDBC and SQLJ to use SSL

To configure connections under the IBM Data Server Driver for JDBC and SQLJ to use SSL, you must set the `DB2BaseDataSource.sslConnection` property to true.

Optional: Set the `DB2BaseDataSource.sslTrustStoreLocation` on a `Connection` or `DataSource` instance to the location of the truststore. Setting the `sslTrustStoreLocation` property is an alternative to setting the Java `javax.net.ssl.trustStore` property. If you set `DB2BaseDataSource.sslTrustStoreLocation`, the `javax.net.ssl.trustStore` property is not used.

You can also set the `DB2BaseDataSource.sslTrustStorePassword` on a `Connection` or `DataSource` instance to identify the truststore password. Setting the `sslTrustStorePassword` property is an alternative to setting the `Java javax.net.ssl.trustStorePassword` property. If you set `DB2BaseDataSource.sslTrustStorePassword`, `javax.net.ssl.trustStorePassword` is not used.

The code sample in Example 24 shows how to set the `sslConnection` property on a `Connection` instance.

Example 24 sslConnection property

```
java.util.Properties props = new java.util.Properties();
props.put( "user", userid );
props.put( "password", passwd );
props.put( "sslConnection", "true" );
java.sql.Connection con =
    java.sql.DriverManager.getConnection(url, props);
```

Configure the Java Runtime Environment to use SSL

Before you can use SSL with your JDBC and SQLJ applications, you need to configure the Java Runtime Environment (JRE) to use SSL. Consider these prerequisites:

- The JRE must include a Java security provider. You must install either the IBM Java Secure Socket Extension (JSSE) provider or the Sun JSSE provider. The IBM JSSE provider is automatically installed with the IBM SDK for Java.

Restriction: You cannot use the Sun JSSE provider with the IBM Java Runtime Environment. If you use the Sun JSSE provider, it only works with the Sun Java Runtime Environment.

- SSL support must be configured at the target server.

We show a configuration for server authentication and a configuration for client authentication.

Configure the JRE to use SSL server authentication

The following steps demonstrate how to set up the JRE to use SSL server authentication:

1. Obtain the server's CA certificate and store it in a Java truststore. One method is to use the FTP command to GET the file in ASCII mode.

Important: Do not use BINARY mode transmission to get the certificate.

Use the Java **keytool** command and specify the **-import** option to import the certificate into the truststore. If you want to create your own truststore, use the **-genkey** option with the Java **keytool** command to create your keystore before you import the certificate. For example, to create a truststore named "myTrustStore", type the following command:

```
keytool -genkey -keystore myTrustStore
```

Keytool interface: In Java Standard Edition (SE) 6 (or later), the keytool interface has changed. The **-genkey** option has been renamed to **-genkeypair**.

Creating your own keystore is an optional step. The default truststore that comes with JSSE is named `<java-home>/lib/security/jssecacerts` or `<java-home>/lib/security/cacerts`. For example, to import the server's CA certificate, "svl224.cacert", to "myTrustStore", type the following command:

```
keytool -import -file svl224.cacert -trustcacerts -keystore myTrustStore -alias svl224_ca
```

If you do not specify the **-keystore** option for the **keytool** command, the default truststore is used.

Keytool interface: In Java SE 6 (or later), the keytool interface has changed. The **-import** option has been renamed to **-importcert**.

2. To verify that the server's CA certificate was imported correctly into the truststore, you can display the contents of the truststore by using the **-list** option for the **keytool** command. For example, to display the contents of the truststore for the alias "svl224_ca", type the following command:

```
keytool -list -alias svl224_ca -keystore myTrustStore -v
```

3. If you have defined your own truststore, you can use these Java system properties to specify the truststore for your Java application to use:
 - `javax.net.ssl.trustStore`: Specifies the name of the customized truststore that you specified with the **-keystore** parameter in the **keytool** utility.

Important: If the IBM Data Server Driver for JDBC and SQLJ property `DB2BaseDataSource.sslTrustStoreLocation` is set, its value overrides the `javax.net.ssl.trustStore` property value.

- `javax.net.ssl.trustStorePassword` (optional): Specifies the password for the truststore. We suggest that you specify a password for the truststore. If not, you cannot protect the integrity of the truststore.

Important: If the IBM Data Server Driver for JDBC and SQLJ property `DB2BaseDataSource.sslTrustStorePassword` is set, its value overrides the `javax.net.ssl.trustStorePassword` property value.

You can set these system properties either statically or dynamically:

- To set these system properties statically, use the **-D** option of the **java** command. For example, to run an application named "testssl.java" and set the `javax.net.ssl.trustStore` system property to specify a truststore named "myTrustStore", type the following:

```
java -Djavax.net.ssl.trustStore=myTrustStore testssl
```

- To set these system properties dynamically, call the `java.lang.System.setProperty` method in your code:

```
System.setProperty("javax.net.ss.trustStore", "myTrustStore");
```

Configure the JRE to use SSL client authentication

The following steps demonstrate how to configure the JRE to use SSL client authentication:

1. Perform all the steps from "Configure the JRE to use SSL server authentication".

2. Use the Java **keytool** command to generate a self-signed CA certificate. For example, to create a self-sign CA certificate in the keystore, “myKeyStore”, issue the following **keytool** command:

```
keytool -genkeypair -alias clnt_cert -keyalg rsa -keystore myKeyStore  
-storepass 123456
```

Follow the prompts to complete the generation of the client certificate.

3. Export the client certificate using the Java **keytool** command. For example, use this command:

```
keytool -exportcert -rfc -alias clnt_cert -file client.ca -keystore myKeyStore  
-storepass 123456
```

The output from this command produces a file called `client.ca`, which contains the client certificate.

4. You might want to verify the client certificate by using the Java **keytool -printcert** command. For example, to verify the exported `client.ca` file, issue the following Java **keytool** command:

```
keytool -printcert -v -file client.ca
```

Figure 9 shows a sample of the output.

```
C:\Dev\JCC>keytool -printcert -v -file client.ca  
Owner: CN=Bob, OU=DB2, O=IM, L=San Jose, ST=CA, C=US  
Issuer: CN=Bob, OU=DB2, O=IM, L=San Jose, ST=CA, C=US  
Serial number: 4dfc050d  
Valid from: Fri Jun 17 18:53:17 PDT 2011 until: Thu Sep 15 18:53:17 PDT 2011  
Certificate fingerprints:  
    MD5:  C5:84:FA:0E:CD:CB:17:9B:B4:1F:0F:4E:90:AA:0C:F4  
    SHA1: EC:4F:88:BC:FA:A8:C7:17:4F:B6:9C:5B:87:6C:8D:2B:8D:9A:CA:2C  
Signature algorithm name: SHA1withRSA  
Version: 3
```

Figure 9 Sample output from the keytool -printcert command

5. Put the client certificate, “client.ca”, at the server and register it with RACF. One method is to use the FTP command to PUT the file (`client.ca`) in ASCII mode.

Important: Do not use BINARY mode transmission to put the certificate.

To register the client certificate with RACF, see “Register client CA certificate with RACF (optional)” on page 22.

6. If you have defined your own keystore, you can use these Java system properties to specify the keystore for your Java application to use:
 - `javax.net.ssl.keyStore`: Specifies the name of the customized keystore that you specified with the **-keystore** parameter in the **keytool** utility.

Important: If the IBM Data Server Driver for JDBC and SQLJ property `DB2BaseDataSource.sslKeyStoreLocation` is set, its value overrides the `javax.net.ssl.keyStore` property value.

- `javax.net.ssl.keyStorePassword` (optional): Specifies the password for the keystore. We suggest that you specify a password for the keystore. If not, you cannot protect the integrity of the keystore.

Important: If the IBM Data Server Driver for JDBC and SQLJ property `DB2BaseDataSource.sslKeyStorePassword` is set, its value overrides the `javax.net.ssl.keyStorePassword` property value.

You can set these system properties either statically or dynamically:

- To set the system property statically, use the `-D` option of the `java` command. For example, to run an application named `testssl.java` and set the system property, `javax.net.ssl.keyStore` for the named “myKeyStore”, issue the following `java` command:

```
java -Djavax.net.ssl.keyStore=myKeyStore testssl
```

- To set the system property dynamically, call the `java.lang.System.setProperty` method in your program:

```
System.setProperty("javax.net.ssl.keyStore", "myKeyStore");
```

Configuring non-Java DB2 clients for Linux, UNIX, and Windows to use SSL

Starting with DB2 Version 9.5 Fix Pack 2, you can configure non-Java DB2 clients for Linux, UNIX, and Windows, such as .NET Data Provider, command-line processor (CLP), and command-line interface (CLI), to use SSL for communications with a DB2 10 for z/OS (or later) server. We describe the following configuration steps in the next sections:

- ▶ Ensuring that the IBM Global Security Kit (GSKit) is available
- ▶ Obtaining the server’s CA certificate and adding it to your client key database
- ▶ Configuring the appropriate client configuration parameter to use SSL

Ensuring that the IBM Global Security Kit (GSKit) is available

For non-Java DB2 clients for the Linux, UNIX, and Windows environments, you use the IBM Global Security Kit (GSKit) to administer digital certificates and key databases. The GSKit is installed automatically on the computer system during the installation of the DB2 Connect™ server. If you do not have the GSKit installed, you can install the GSKit libraries from the IBM DB2 Support Files for SSL Functionality DVD. Alternatively, you can install the GSKit libraries from an image that you download from Passport Advantage®.

For information about the GSKit tool `GSKCapiCmd`, see the *GSKCapiCmd User’s Guide*, which is available at the following Web page:

ftp://ftp.software.ibm.com/software/webserver/appserv/library/v61/ihg/GSK7c_CapiCmd_UserGuide.pdf

You also need to ensure that these environment settings are configured, depending on the client platform:

- ▶ On a Windows (32-bit or 64-bit) platform, verify that the path to the GSKit libraries is included in the `PATH` environment variable. For example, on Windows, add the GSKit `bin` and `lib` directories to the `PATH` environment variable:

```
set PATH="C:\Program Files\IBM\gsk7\bin";"C:\Program Files\IBM\gsk7\lib";%PATH%
```

or

```
set PATH="C:\Program Files\IBM\gsk8\bin";"C:\Program Files\IBM\gsk8\lib";%PATH%
```

- On Linux and UNIX platforms, the GSKit libraries are located in `sqllib/lib` or `sqllib/lib64`. To set the `LIBPATH`, `SHLIB_PATH`, or `LD_LIBRARY_PATH` environment variable, you issue the following commands:

```
export LIBPATH=$INSTHOME/sqllib/lib:$LIBPATH:.
export SHLIB_PATH=$INSTHOME/sqllib/lib:$SHLIB_PATH:.
export LD_LIBRARY_PATH=$INSTHOME/sqllib/lib:$LD_LIBRARY_PATH:.
```

Alternatively, if you want to use the IBM Key Management tool, `gsk7ikm`, you need to set the `JAVA_HOME` environment variable before you can start it. For information about how to navigate the IBM Key Management tool to create a client key database and add the server's CA certificate to the client key database, refer to *DB2 9 for z/OS: Distributed Functions*, SG24-6952-01.

The SSL communication must always be in the Federal Information Processing Standard (FIPS) mode.

Obtaining the server's CA certificate and adding it to your client key database

To prepare a DB2 client for Linux, UNIX, and Windows to access DB2 10 for z/OS using SSL, you need to perform the following tasks:

1. Obtain the server's CA certificate from the target DB2 server. One method is to use the FTP command to GET the file in ASCII mode from the server computer.

Important: Do not use BINARY mode transmission to get the certificate.

2. On the DB2 client computer, use the `gsk7capicmd` tool to create a client key database of the Certificate Management System (CMS) type. The `gsk7capicmd` tool is a non-Java-based command-line tool. Therefore, it does not require Java to be installed on your computer to run it.

For example, to create a client key database named "mydbclnt.kdb" with a password of "myClnPw0rd" of type CMS in the directory `$DB2PATH\security\keystore`, use the following `gsk7capicmd` command:

```
gsk7capicmd -keydb -create -db "mydbclnt.kdb" -pw "myClnPw0rd" -type CMS
-stash -fips
```

Specifying the `-stash` option creates a stash file for storing the password to the key database after creation. A *stash file* is an automatic way of providing the password to access the key database. Also, the password is encrypted when stored in the stash file so that when access to the key database is required, the system decrypts the password from the stash file and uses it as input to access the key database. The stash file has the following format:

```
<key database>.sth
```

You must specify the location of the stash file when accessing the key database.

3. Use the `gsk7capicmd` command to add the server's CA certificate to the key database that was created in step 2.

For example, use the following command to add the server's CA certificate from the file `ec754.cacert` into the key database named "mydbclnt.kdb":

```
gsk7capicmd -cert -add -db "mydbclnt.kdb" -pw "myClntPw0rd" -file  
"ec754.cacert" -label "EC754 Server CA" -format ascii
```

Important: The label name that is specified for the **-label** option must match the label that is attached to the server's CA certificate. If not, the SSL connection is not successfully established.

To verify that the server's CA certificate was added into the key database successfully, issue the following **gsk7capicmd** command:

```
gsk7capicmd -cert -details -db "mydbclnt.kdb" -pw "myClntPw0rd" -label "EC754  
Server CA"
```

Configuring the appropriate client configuration parameter to use SSL

If your DB2 client level is at Version 9.7 or later, you do not need to create an SSL configuration file. DB2 Version 9.7 introduced two new DB2 configuration parameters (**ssl_clnt_keydb** and **ssl_clnt_keystash**) to replace the SSL parameters (**DB2_SSL_KEYSTORE_FILE** and **DB2_SSL_KEYRING_STASH_FILE**) that were used for specifying the fully qualified path for the client key database and for the stash file. So, you can skip to step 3. Otherwise, you need to create an SSL configuration file:

1. Create an SSL configuration file.

Depending on the platform type, the SSL configuration file needs to be stored in a specific directory on the client system:

- For Linux and UNIX: *\$INSTHOME/cfg*
- For Windows: *\$INSTHOME/*

An example of *\$INSTHOME* on the Windows platform is a concatenation of the *\$DB2INSTPROF/\$DB2INSTDEF* environment variables:

```
C:\DOCUMENTS AND SETTINGS\ALL USERS\APPLICATION DATA\IBM\DB2\DB2COPY1\DB2
```

- Name the SSL configuration file, **SSLClientconfig.ini**

2. Define the following SSL parameters in the **SSLClientconfig.ini** file:

- **DB2_SSL_KEYSTORE_FILE**: Specifies the fully qualified name of the key database that stores the server's CA certificate.
- **DB2_SSL_KEYRING_STASH_FILE**: Specifies the fully qualified name of the stash file that contains the encrypted password to access the key database.

The following example shows an **SSLClientconfig.ini** file:

```
DB2_SSL_KEYSTORE_FILE=C:\Program  
Files\IBM\SQLLIB\security\keystore\mydbclnt.kdb  
DB2_SSL_KEYRING_STASH_FILE=C:\Program  
Files\IBM\SQLLIB\security\keystore\mydbclnt.sth
```

3. Set the appropriate connection strings or configuration parameters for your client application:

- CLP and embedded SQL clients

For CLP clients, you need to issue the **CATALOG TCPIP NODE** statement with the **SECURITY** keyword set to "SSL" to specify SSL for the connection, as shown in the following example:

```
db2 catalog tcpip node tcp754as remote fvttec754.svldev.svl.ibm.com server  
448 security ssl
```

```
db2 catalog db db754as as stl754as at node tcp754as authentication server
```

```
db2 catalog dcs db db754as as stlec1
```

Important: The port value for the SERVER keyword must identify the target server's secure port. If not, the connection fails and error SQL30081N is returned.

If you use DB2 Version 9.7 or later, define the `ssl_clnt_keydb` and `ssl_clnt_stash` configuration parameters with the fully qualified name of the client key database (.kdb) and the fully qualified name of the client key database stash file (.sth) respectively.

Update the Database Manager Configuration, as shown in the following example:

```
db2 update dbm cfg using SSL_CLNT_KEYDB "c:\program
files\ibm\sqllib\security\keystore\mydbc1nt.kdb" SSL_CLNT_STASH "c:\program
files\ibm\sqllib\security\keystore\mydbc1nt.sth"
```

Connect to the server using SSL from the CLP client:

```
db2 connect to stl754as user <userid> using <password>
```

Alternatively, an embedded SQL client can use the following statement to connect to the server:

```
strcpy(dbAlias,"stl754as");
EXEC SQL CONNECT TO :dbAlias USER :userid USING :pwd
```

– CLI/Open Database Connectivity (ODBC) applications

Depending on which environment your CLI application executes, you can use either the connection string parameters (`ssl_client_keystoredb` and `ssl_client_keystash`) or DB2 configuration parameters (`ssl_clnt_keydb` and `ssl_clnt_stash`) to specify the fully qualified path to the client key database and the stash file. Connection string parameters are only applicable to DB2 Version 9.7 or later clients.

For example, if your CLI application uses the IBM Data Server Driver for ODBC and CLI, and the DB2 client level is Version 9.7 or later, use the connection string parameters to specify the client key database and stash file.

Call `SQLDriverConnect` with a connection string that contains the `SECURITY=SSL` keyword:

```
"Database=stl754as; Protocol=tcpip; Hostname=fvtec754.svldev.svl.ibm.com;
Servicename=448; Security=ssl; Ssl_client_keystoredb=c:\program
files\ibm\sqllib\security\keystore\mydbc1nt.kdb;
Ssl_client_keystash=c:\program
files\ibm\sqllib\security\keystore\mydbc1nt.sth;"
```

For example, if your CLI application uses the IBM Data Server Client or IBM Data Server Runtime Client, and the DB2 client level is Version 9.7 or later, you can use either the connection string parameters or DB2 configuration parameters to specify the fully qualified path to the client key database and stash file.

To set the connection string parameter in the `db2cli.ini` file:

```
[stl754as]
Database=stl754as
Protocol=tcpip
Hostname=fvtec754.svldev.svl.ibm.com
Servicename=448
Security=ssl
SSL_client_keystoredb=c:\program
files\ibm\sqllib\security\keystore\mydbc1nt.kdb
```

```
SSL_client_keystash= c:\program  
files\ibm\sqllib\security\keystore\mydbc1nt.sth
```

Parameters: If the connection string parameters are used, these values override the values of the DB2 configuration parameters.

- DB2 Data Provider for .NET applications

A DB2 Data Provider for .NET Framework application can establish an SSL connection to a remote server by specifying the connection string keyword SECURITY with the value “SSL” to use SSL for the connection to the remote server. With DB2 Version 9.7, you can also specify the fully qualified path for the client key database file (.kdb) and for the stash file (.sth) using the SSLClientKeystoredb and SSLClientKeystash connection string parameters.

Configuring remote client applications to use SSL through a DB2 Connect server for Linux, UNIX, and Windows

In this section, we discuss the configuration steps to set up remote client applications (Java applications or non-Java DB2 applications) to use SSL to access a DB2 for z/OS server through a DB2 Connect server for Linux, UNIX, and Windows.

In certain client environments, remote client applications that need to access a DB2 for z/OS server system can only do so through a DB2 Connect server. Furthermore, the remote client applications might be required to use SSL to connect to the DB2 Connect server, because data transmission can occur over an open network. However, if the DB2 Connect server and the DB2 for z/OS server communicate with each other over a closed network (that is, behind a secure firewall), the connections between the DB2 Connect server and the DB2 for z/OS server do not necessarily require the use of SSL. The network is protected by a firewall or another similar secure network technology. As a result, in this type of environment, we need to perform the following configuration steps:

- ▶ Configuring SSL support in a DB2 Connect server for Linux, UNIX, and Windows
- ▶ Configuring Java or non-Java applications to use SSL to access the DB2 Connect server

Configuring SSL support in a DB2 Connect server for Linux, UNIX, and Windows

In our scenario, we configure the DB2 Connect server to support SSL to create a secure connection between the remote client and the server. The connection between the DB2 Connect server and the DB2 for z/OS server can remain as unsecured if the network operates behind a secure firewall.

In a DB2 Connect server for Linux, UNIX, and Windows environment, you use the IBM Global Security Kit (GSKit) to administer digital certificates and key databases. The GSKit is installed automatically on the computer system during the installation of the DB2 Connect server. If you do not have the GSKit installed, you can install the GSKit libraries from the IBM DB2 Support Files for SSL Functionality DVD. Alternatively, you can install the GSKit libraries from an image that you have downloaded from Passport Advantage.

For information about the GSKit tool GSKCapiCmd, see the *GSKCapiCmd User's Guide*, which is available at the following Web page:

ftp://ftp.software.ibm.com/software/webserver/appserv/library/v61/ihs/GSK7c_CapiCmd_UserGuide.pdf

Before we can proceed, ensure that the following environment settings are configured:

- On Windows (32-bit and 64-bit) platforms, verify that the path to the GSKit libraries is included in the *PATH* environment variable. For example, on Windows, add the GSKit bin and lib directories to the *PATH* environment variable:

```
set PATH="C:\Program Files\IBM\gsk7\bin";"C:\Program Files\IBM\gsk7\lib";%PATH%
```

or

```
set PATH="C:\Program Files\IBM\gsk8\bin";"C:\Program Files\IBM\gsk8\lib";%PATH%
```

On Linux and UNIX platforms, the GSKit libraries are located in *sql1lib/lib* or *sql1lib/lib64*. So, to set the *LIBPATH*, *SHLIB_PATH*, or *LD_LIBRARY_PATH* environment variable, you issue the following commands:

```
export LIBPATH=$INSTHOME/sql1lib/lib:$LIBPATH:.
```

```
export SHLIB_PATH=$INSTHOME/sql1lib/lib:$SHLIB_PATH:.
```

```
export LD_LIBRARY_PATH=$INSTHOME/sql1lib/lib:$LD_LIBRARY_PATH:.
```

- Ensure that the connection concentrator feature is disabled. SSL support is not enabled in the DB2 instance if the connection concentrator feature is enabled.

To determine if the connection concentrator feature is enabled, issue the *GET DATABASE CONFIGURATION* command. If the configuration parameter **max_connections** is set to a value greater than the value of **max_coordagents**, the connection concentrator feature is enabled.

To configure SSL support for a DB2 Connect server, you need to create a server key database to manage the digital certificates. Next, the DB2 instance owner must configure the DB2 instance for SSL support:

1. Create a server key database and the associated digital certificates using the GSKCapiCmd tool:
 - a. Use the GSKCapiCmd tool to create a server key database of the type CMS. The GSKCapiCmd tool is a non-Java-based command-line tool. Java does not need to be installed on your system to use this tool.

The path to the GSKCapiCmd tool is *C:\Program Files\IBM\gsk7\bin* (for DB2 V9.5 FP2) or *C:\Program Files\IBM\gsk8\bin* (for DB2 V9.7) on a 32-bit and 64-bit Windows platform. On 64-bit platforms, the 32-bit GSKit executable files and libraries are also present; in which case, the path for the command is *C:\Program Files (x86)\IBM\GSK8\bin*, and *sql1lib/gskit/bin* on Linux and UNIX platforms.

For example, to create a key database called *myserver.kdb* and a stash file called *myserver.sth*, issue the following **GSKCapiCmd** command:

```
gsk8capicmd -keydb -create -db "myserver.kdb" -pw "myServ3rPwd" -type cms -stash
```

Specifying the **-stash** option creates a stash file for storing the password to the key database after creation. A stash file is an automatic way of providing the password to access the key database. The password is also encrypted when stored in the stash file so that when access to the key database is required, the system decrypts the password from the stash file and uses it as input to access the key database.

- b. Add a certificate to the server key database that we created in step a. This certificate is used by the remote clients to authenticate the server during the SSL handshake phase. You can obtain a certificate by using the GSKCapiCmd tool to create a certificate

request and submit it to a Certificate Authority (CA) to be signed, or you can create a self-signed certificate for testing purposes.

For example, to create a self-signed certificate with a certificate label of "myselfsigned", issue the following **GSKCapiCmd** command:

```
gsk8capicmd -cert -create -db "myserver.kdb" -pw "myServ3rPwd" -label  
"myselfsigned" -dn  
"CN=myhost.mydomain.com,O=myOrganization,OU=myOrganizationUnit,L=myLocation,  
ST=myState,C=myCountry"
```

- c. Extract the server certificate from the server key database and distribute this certificate to all of the remote clients that connect to this server using SSL.

For example, to extract the server self-signed certificate that you created from the prior step, issue the following **GSKCapiCmd** command:

```
gsk8capicmd -cert -extract -db "myserver.kdb" -pw "myServ3rPwd" -label  
"myselfsigned" -target "myserver.arm" -format ascii -fips
```

The extracted certificate is saved in the file called `myserver.arm`.

2. Configure the DB2 Connect server for SSL support. You might need to log in as the DB2 instance owner before you can set the SSL configuration parameters and DB2COMM registry variables.

- a. Specify the fully qualified path of the server key database in the `SSL_SVR_KEYDB` database configuration parameter, for example:

```
db2 update dbm cfg using SSL_SVR_KEYDB C:\SSLServer\myserver.kdb
```

- b. Specify the fully qualified path of the server key database stash file in the `SSL_SVR_STASH` database configuration parameter, for example:

```
db2 update dbm cfg using SSL_SVR_STASH C:\SSLServer\myserver.sth
```

SSL_SVR_STASH: If the `SSL_SVR_STASH` database configuration parameter is NULL (unspecified), SSL support is disabled.

- c. Specify the label of the server certificate in the `SSL_SVR_LABEL` database configuration parameter, for example:

```
db2 update dbm cfg using SSL_SVR_LABEL myselfsigned
```

SSL_SVR_LABEL: If the `SSL_SVR_LABEL` database configuration parameter is NULL (unspecified), the default certificate label in the server key database is used. If there is no default certificate label in the server key database, SSL support is disabled.

- d. Specify the secure port number for the DB2 Connect server to accept secure connections in the `SSL_SVCENAME` database configuration parameter, for example:

```
db2 update dbm cfg using SSL_SVCENAME 50088
```

Important: If the `DB2COMM` registry variable specifies both TCP/IP and SSL (that is, `DB2COMM=TCP,SSL`), the port number that you specify for the `SSL_SVCENAME` must differ from the port number that is associated with `SVCENAME`. If the port number that you specify for the `SSL_SVCENAME` is the same as the port number for the `SVCENAME`, neither TCP/IP nor SSL support is enabled. Also, if the `SSL_SVCENAME` is set to NULL (unspecified), SSL support is disabled.

- e. (Optional) You can specify a cipher suite for the server to use for SSL by setting the `SSL_CIPHERSPECS` database configuration parameter. We suggest that you leave this SSL database configuration parameter unspecified so that the GSKit can select the strongest available cipher suite that is supported by both the client and server.

GSKit supports the following cipher suites:

- `TLS_RSA_WITH_AES_256_CBC_SHA`
- `TLS_RSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_3DES_EDE_CBC_SHA`

- f. Set the SSL communication protocols for the DB2 instance in the `DB2COMM` registry variable. For example, use the `db2set` command to set `DB2COMM` with the value of `TCPIP` and `SSL`:

```
db2set -i db2instdef DB2COMM=TCPIP,SSL
```

In this example, `db2instdef` is the DB2 instance name. If you want to start DB2 with only SSL support, you can set `DB2COMM=SSL`.

- g. Restart the DB2 instance, for example:

```
db2stop  
db2start
```

Configuring Java or non-Java applications to use SSL to access the DB2 Connect server

To configure Java applications using the IBM Data Server Driver for JDBC and SQLJ (Type 4 connections) to use SSL, refer to “Configure connections under the IBM Data Server Driver for JDBC and SQLJ to use SSL” on page 30.

To configure non-Java DB2 clients for Linux, UNIX, and Windows to use SSL, refer to “Configuring non-Java DB2 clients for Linux, UNIX, and Windows to use SSL” on page 34.

For either type of client, you need to import the server’s certificate, the `myserver.arm` file, that you created from the previous step to your client keystore or key database. You also need to specify the correct port number of the server that accepts secure connections.

Client access to DB2 using digital certificates

With DB2 10 for z/OS, a remote client application can access a DB2 10 for z/OS server using the SSL client authentication (which is discussed in “Coding the AT-TLS policy rules for SSL client authentication” on page 17 and “Register client CA certificate with RACF (optional)” on page 22) security protocol.

Note: To enable this support, APAR PM37057 (PTF UK73180) needs to be installed.

After PTF UK73180 is installed, the remote client application can securely access DB2 using a user ID only knowing that the identity of the client is properly authenticated by the server that was configured with SSL client authentication.

Restriction: Currently, non-Java clients for Linux, UNIX, and Windows do not support this feature.

Today, the clients that support client access using digital certificates are the IBM Data Server Driver for JDBC and SQLJ, and DB2 for z/OS requester.

Before we proceed with the steps to configure the client and server to use the client login using digital certificates, we briefly explore the DB2 10 for z/OS processing to authenticate a remote client accessing DB2 using digital certificates.

Processing client access using digital certificates at a DB2 10 for z/OS server

The DB2 server completes a sequence of authentication tasks when handling a remote client connection request, as shown in Figure 10.

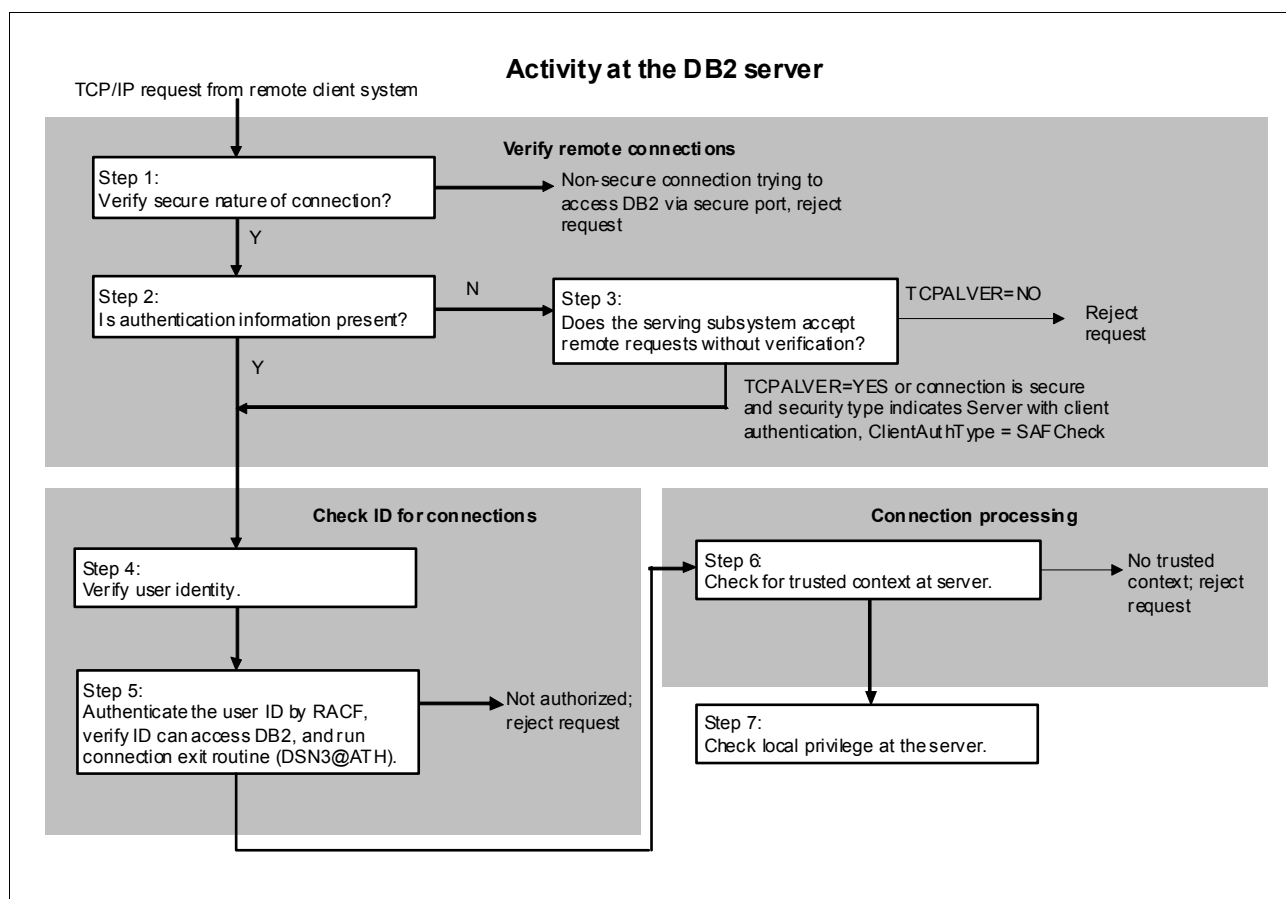


Figure 10 Processing of inbound TCP/IP connection request at a DB2 server

The following numbers correspond to the steps in Figure 10:

1. If the connection is accepted to a secure port, the secure nature of the connection must be verified. In order for DB2 to determine if the connection is secure, DB2 issues the AT-TLS SIOCTTLSCCTL IOCTL command to query the AT-TLS policy information for the connection:
 - If the IOCTL returns a policy status of TTLS_POL_ENABLED and a connection status of TTLS_CONN_SECURE, a matching policy rule is found and SSL is enabled for the connection. Furthermore, if the policy type is configured with the HandshakeRole parameter set to ServerWithClientAuth and the ClientAuthType parameter is set to SAFCheck, the IOCTL also returns the user ID that is associated with the (client) certificate, which is registered with RACF.

- If the IOCTL returns a policy status of TTLS_POL_NO_POLICY, a matching policy rule is not found for the connection, or the IOCTL returns a policy status of TTLS_POL_NOT_ENABLED, a matching rule policy is found for the connection but a policy is not configured to allow a secure connection for that client. In both cases, the connection is not secure.

If the connection is not secure and the remote connection is trying to access DB2 via the secure port, DB2 will reject the connection request.

- DB2 checks to see if an authentication token (RACF-encrypted password, RACF PassTicket, DRDA-encrypted password, or Kerberos ticket) accompanies the remote request.
- If no authentication token is supplied, DB2 checks the TCPALVER subsystem parameter to see if DB2 accepts IDs without authentication information:
 - If TCPALVER=NO | SERVER, DB2 requires the minimum of a user ID and a password.
 - If TCPALVER=SERVER_ENCRYPT, DB2 requires a user ID and a password. In addition, it requires that the security credentials are AES-encrypted or that the connection is accepted on a port that ensures AT-TLS policy protection, such as a DB2 Security Port (SECPORT). Kerberos tickets are accepted. RACF PassTickets or non-encrypted security credentials are accepted only when the connection is secured by the TCP/IP network.
 - If TCPALVER=YES | CLIENT, DB2 accepts TCP/IP connection requests that contain only a user ID.
 - If the connection is secure (which was discussed in step 1 on page 42) and the user ID that is associated with the client certificate is known, DB2 also accepts TCP/IP connection requests that contain only a user ID.
- DB2 invokes RACF to verify the user ID. The user identity (ID) is one of the following values:
 - The ID is a Kerberos principal that is validated by the Kerberos Security Server, if a Kerberos ticket is provided.
 - The ID is a RACF ID that is authenticated by RACF together with the password or PassTicket, if provided.
 - The ID is a distributed ID if the presence of a distributed registry name is provided by the connection. When a distributed registry name is provided, the RACF user ID is derived from a distributed name filter that is defined in RACF. To use distributed name filters, in this situation, DB2 invokes RACF's distributed identity propagation function to retrieve the RACF user ID that is associated with the distributed identity and registry name. To use distributed identity propagation, ensure that you define a distributed identity filter, which maps the distributed identity and the distributed registry name with a RACF user ID.

If RACF is unable to locate a distributed identity name filter for the distributed identity and registry name pair, DB2 will treat the distributed identity as a RACF user ID and invoke RACF to verify it together with the password, if provided.
- The user ID is verified and authenticated in the following manner:
 - When Kerberos tickets are used, the RACF ID is derived from the Kerberos principal identity. To use Kerberos tickets, ensure that you map Kerberos principal names with RACF IDs.
 - When the remote client accesses DB2 via a secure connection and a RACF user ID is associated with the client certificate, DB2 compares the remote client user ID with the certificate user ID.

- Assuming that the remote client user ID and the user ID that is associated with the client certificate differ, additional authentication processing by DB2 is required to authenticate the remote client accessing DB2 using a certificate.

Trusted connection: If DB2 processes a trusted context switch-user request, DB2 is not required to give any special consideration to authenticate the remote client user ID, because the connection has already been established as a trusted connection.

- If the remote client user ID and the user ID that is associated with the client certificate are the same, DB2 authenticates the user ID (and password, if provided) with RACF, normally.

Important: Additional authentication processing is required to authenticate the remote client accessing DB2 using a certificate, as explained next.

Additional authentication is required to authenticate the remote client connection when the remote connection request accesses DB2 using a digital certificate and the remote client user ID differs from the RACF user ID that is associated with the client certificate. DB2 performs additional steps:

- The user ID that is associated with the client certificate is authenticated by RACF. This user ID is also verified by RACF to ensure that it can access DB2 and the connection exit routine is invoked.
- Assuming the authentication is successful, DB2 searches for a matching trusted context that is defined for the user ID that is associated with the client certificate. If the trusted context exists, an implicit trusted connection is now established between the remote connection request and the DB2 server.

When the trusted connection is established, DB2 authenticates the remote client user ID that is associated with the remote connection request. This user ID is also verified by RACF to ensure that it can access DB2 and the connection exit routine is invoked.

Assuming that the authentication is successful, DB2 determines if the primary user ID is allowed to use the trusted connection. If the trusted connection can be used, the remote client is allowed access into DB2 using a digital certificate.

If authentication is unsuccessful for the remote client user ID, or the remote client user ID is not allowed to use the trusted connection, the remote connection request is rejected.

- If authentication is unsuccessful for the client certificate user ID, or a trusted connection was not able to be established for the client certificate user ID, the remote connection is rejected.

If special consideration to authenticate the remote client connection request is not required, the user ID together with the password or RACF PassTicket, if provided, is authenticated by RACF. DB2 calls RACF to check the user ID's authorization against the ssnm.DIST resource and invokes the DB2 connection exit routine (DSN3@ATH). The parameter list that is passed to the routine describes where the remote request originated.

In addition, depending on your RACF environment, the following RACF checks might also be performed:

- If the RACF APPCPORT class is active, RACF verifies that the ID is authorized to access z/OS from the port of entry (POE). The POE that RACF uses in the RACROUTE VERIFY call depends on whether all the following conditions are true:
 - The current operating system is z/OS V1.5 or later.

- The TCP/IP Network Access Control is configured.
- The RACF SERVAUTH class is active.

If all these conditions are true, RACF uses the remote client's POE security zone name that is defined in the TCP/IP Network Access Control file. If one or more of these conditions are not true, RACF uses the literal string 'TCPIP'.

If this request is to change a password, the password is changed.

6. The remote request has a primary authorization ID, possibly one or more secondary IDs, and an SQL ID. (The SQL ID cannot be translated.) If the remote connection request originated from a DB2 for z/OS client, the plan or package owner ID also accompanies the request. Privileges and authorities that are granted to those IDs at the DB2 server govern the actions that the request can take.

DB2 searches for a matching trusted context. If DB2 finds a matching trusted context, it validates the following attributes:

- If the SERVAUTH attribute is defined for the identified trusted context and TCP/IP provides a RACF SERVAUTH profile name to DB2 during the establishment of the connection, DB2 matches the SERVAUTH profile name with the SERVAUTH attribute value.
- If the SERVAUTH attribute is not defined or the SERVAUTH name does not match the SERVAUTH that is defined for the identified trusted context, DB2 matches the remote client's TCP/IP address with the ADDRESS attribute that is defined for the identified trusted context.
- If the ENCRYPTION attribute is defined, DB2 validates whether the connection uses the proper encryption, as specified in the value of the ENCRYPTION attribute.
- If the DEFAULT SECURITY LABEL attribute is defined for the system authorization ID, DB2 verifies the security label with RACF. This security label is used for verifying multilevel security for the system authorization ID. However, if the system authorization ID is also in the ALLOW USER clause with SECURITY LABEL, that ID is used.

If the validation is successful, DB2 establishes the connection as trusted. If the validation is not successful, the connection is established as a normal connection without any additional privileges, DB2 returns a warning, and SQLWARN8 is set.

When the trusted connection is established, DB2 enables the trusted connection to be reused by a separate user ID on a transaction boundary. A sequence of tasks is then performed by DB2 when the remote connection requests to switch the user ID on a trusted connection:

- DB2 determines if the primary authorization ID is allowed to use the trusted connection. If the WITH AUTHENTICATION clause is specified for the user, DB2 requires an authentication token for the user. The authentication token can be a password, a RACF passticket, or a Kerberos ticket.
- Assuming that the primary authorization ID is allowed, DB2 determines the trusted context for any SECURITY LABEL definition. If a specific SECURITY LABEL is defined for this user, it becomes the SECURITY LABEL for this user. If no specific SECURITY LABEL is defined for this user but a DEFAULT SECURITY LABEL is defined for the trusted context, DB2 verifies the validity of this SECURITY LABEL for this user through RACF by issuing the RACROUTE VERIFY request.

If the primary authorization ID is allowed, DB2 performs a connection initialization. This connection initialization results in an application environment that truly mimics the environment that is initialized if the new user establishes the connection in the normal DB2 manner. For example, any open cursor is closed, and temporary table information is dropped.

- If the primary authorization ID is not allowed to use the trusted connection or if the SECURITY LABEL verification fails, the connection is returned to an unconnected state. The only operation that is allowed is to establish a valid authorization ID to be associated with the trusted connection. Until a valid authorization is established, if any SQL statement is issued, an error (SQLCODE -900) is returned.

Configuring client access to DB2 using digital certificates

To implement client access to a DB2 10 for z/OS server using digital certificates, you need to perform the following steps:

- ▶ Configure the AT-TLS policy rules for the DB2 server to use SSL client authentication (which was discussed in “Coding the AT-TLS policy rules for SSL client authentication” on page 17 and “Register client CA certificate with RACF (optional)” on page 22).
- ▶ Optionally, create a trusted context definition for the client certificate RACF user ID as the system authorization ID and allow any DRDA user ID to be used by the trusted context.

Example 25 shows how to create a trusted context for the user ID, USRT001, which is associated with the registered client certificate, and allow user IDs, USRT020, USRT021, and ADMF002, to use this trusted context.

Example 25 Sample trusted context definition for client access to DB2 using certificates

```
CREATE TRUSTED CONTEXT CTX1
BASED UPON CONNECTION USING SYSTEM AUTHID USRT001
ATTRIBUTES (ADDRESS '9.30.223.20')
NO DEFAULT ROLE
ENABLE
WITH USE FOR USRT020,USRT021,ADMF002;
```

DB2 is now ready to accept remote client access using a digital certificate that is registered to RACF, which has a user ID, USRT001, that is associated with it.

User ID: With client access to DB2 using a digital certificate, the user ID that the client application specifies for the connection must also be a RACF-defined user ID. It does not have to be the same user ID as the certificate user ID, but the user ID must be known to RACF and have access to DB2 resources (DSNR class).

In the case where the user ID that is specified by the remote client application is a distributed user ID, which is registered to a distributed registry name (that is, the user ID is not defined in RACF), you must configure DB2 support for z/OS identity name filters.

Configuring z/OS identity name filters for the DB2 server

A *distributed identity filter* is a RACF mapping association between a RACF user ID and one or more distributed user identities. You can use the RACF RACMAP command to associate a distributed user identity with a RACF user ID. RACF distributed identity filters are implemented through z/OS identify propagation. You must install and run z/OS Version 1 Release 11 to use distributed identity filters.

DB2 provides support for z/OS identify propagation and distributed identity filters. You need to create distributed identity filters to take advantage of this support. Follow these steps to create a distributed identity filter:

1. Activate the RACF general resource IDIDMAP class and enable it for RACLIST processing by issuing the following command:
 SETROPTS CLASSACT(IDIDMAP) RACLIST(IDIDMAP)

2. Define a distributed identity filter and associate the distributed user name with a RACF user ID by issuing the RACF RACMAP command. To define a filter for a non-LDAP user name, specify the user name as a simple character string to be defined in a non-LDAP registry. Suppose that the distributed user name is 'MARY', which is defined in user registry 'Registry01'. If you want to map this user name to RACF user ID 'USRT021', you can issue the following RACMAP command:

```
RACMAP ID(USRT021) MAP -  
USERIDFILTER(NAME('MARY')) -  
REGISTRY(NAME('Registry01')) -  
WITHLABEL('Filter for MARY from Registry01')
```

3. Refresh the IDIDMAP class profile by issuing the following command:

```
SETOPTS RACLIST(IDIDMAP) REFRESH
```

4. If necessary, review the distributed identity filter by issuing the following RACMAP LISTMAP command:

```
RACMAP ID(USRT021) LISTMAP
```

If the new filter is successfully created, the following output is returned:

```
Mapping information for user USRT021:  
Label: Filter for MARY from Registry01  
Distributed Identity User Name Filter:  
  >MARY<  
Registry name:  
  >Registry01<
```

The new filter assigns RACF user ID, USRT021, when the distributed identity is user MARY from Registry01. When user MARY authenticates her identity at her distributed application server and performs tasks that access a remote DB2 server system, DB2 passes distributed user name MARY and registry name Registry01 as character strings to RACF.

During DB2 remote connection processing, DB2 calls the RACF RACROUTE REQUEST=VERIFY ENVIR=CREATE macro service. RACF uses these data values to search the IDIDMAP profiles for a matching filter. RACF finds the matching filter labeled 'Filter for MARY from Registry01' and assigns it the USRT021 user ID. Because the remote client accesses DB2 using a digital certificate, DB2 performs a switch user with the trusted context, CTX1, to allow USRT021 to become the primary authorization ID for the connection. The remote connection then executes its transactions with the authority of the USRT021 user ID. If in place, audit records for this transaction contain RACF user ID USRT021, distributed user MARY, and registry name Registry01, which DB2 passes to RACF.

References

For additional information, refer to these publications:

- ▶ *Security Functions with IBM DB2 10 for z/OS*, SG24-7959-00
- ▶ *DB2 9 for z/OS: Distributed Functions*, SG24-6952-01
- ▶ *IBM z/OS V1R10 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7696-00
- ▶ *IBM z/OS V1R10 Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7697-00
- ▶ *DB2 10 for z/OS Administration Guide*, SC19-2968-04

- ▶ *z/OS V1R11 Communications Server: IP Configuration Guide*, SC31-8775-15
- ▶ *z/OS V1R11 Communications Server: IP Configuration Reference*, SC31-8776-16
- ▶ *z/OS V1R11 Communications Server: IP Diagnosis Guide*, GC31-8782-10
- ▶ *z/OS V1R11 Security Server RACF Command Language Reference*, SA22-7687-14
- ▶ *z/OS V1R11 Security Server RACF Security Administrator's Guide*, SA22-7683-13
- ▶ *z/OS V1R11 System SSL Programming, Software Dependencies*, SC24-5901-08
- ▶ *DB2 Version 9.5 for Linux, UNIX, and Windows Database Security Guide*, SC23-5850-02
- ▶ *IBM DB2 9.7 for Linux, UNIX, and Windows Developing Java Applications*, SC27-2446-02
- ▶ *IBM DB2 9.7 for Linux, UNIX, and Windows Database Security Guide*, SC27-2443-01

The team who wrote this IBM Redpaper

This paper was produced by a team of specialists working at the Silicon Valley Laboratory, San Jose.

Derek Tempongko is an IBM Advisory Software Engineer, DB2 for z/OS distributed database development, working in the IBM Silicon Valley Laboratory. Derek has eleven years of experience in the DB2 for z/OS distributed database development field. His areas of expertise include distributed database development and security implementation for DB2 remote connectivity. Derek holds a Bachelor's of Science degree in Computing Science from the University of Technology, Sydney, Australia.

Paolo Bruni is an ITSO Project Leader based in IBM Silicon Valley Laboratory. Paolo has authored several IBM Redbooks® publications and Redpapers™ documents about DB2 for z/OS.

Thanks to the following people for their contributions to this project:

Keith Howell
Jim Pickel
High Smith
IBM Silicon Valley Laboratory, San Jose

Now you can become a published author, too!

Here is an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document REDP-4799-00 was created or updated on December 5, 2011.



Send us your comments in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:
ibm.com/redbooks
- Send your comments in an email to:
redbooks@us.ibm.com
- Mail your comments to:
IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400 U.S.A.




Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

DB2 Connect™
DB2®
DRDA®
IBM®

MVS™
Passport Advantage®
RACF®
Redbooks®

Redpaper™
Redpapers™
Redbooks (logo) ®
z/OS®

The following terms are trademarks of other companies:

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.